In This Issue: IBM LinkWay

$4.95

HYPERLINK MAGAZINE

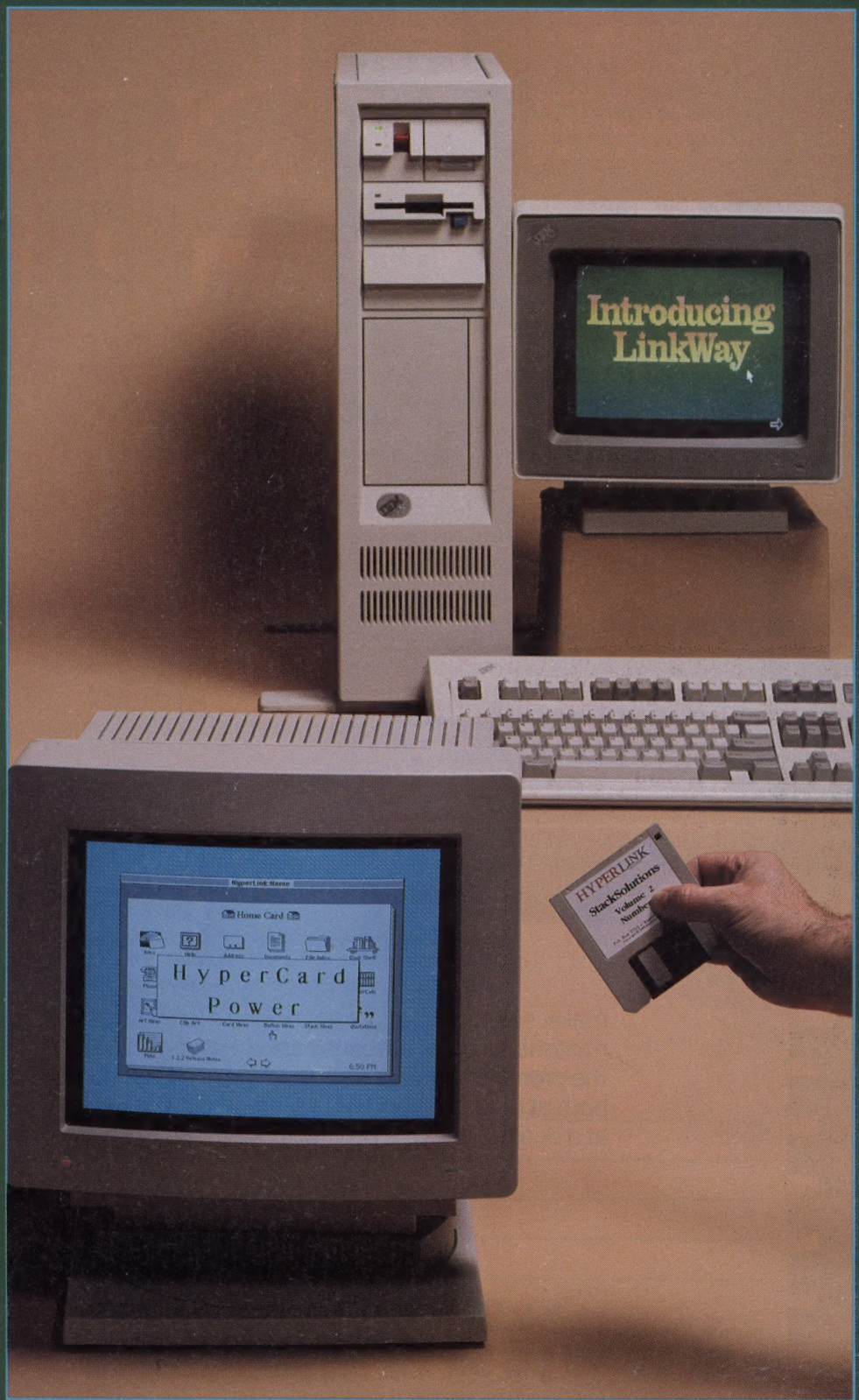
May/June 1989 • Vol. 2 No. 3

A Neural Network Simulated in HyperCard

A Depreciation Calculator in HyperCard for Fixed Assets

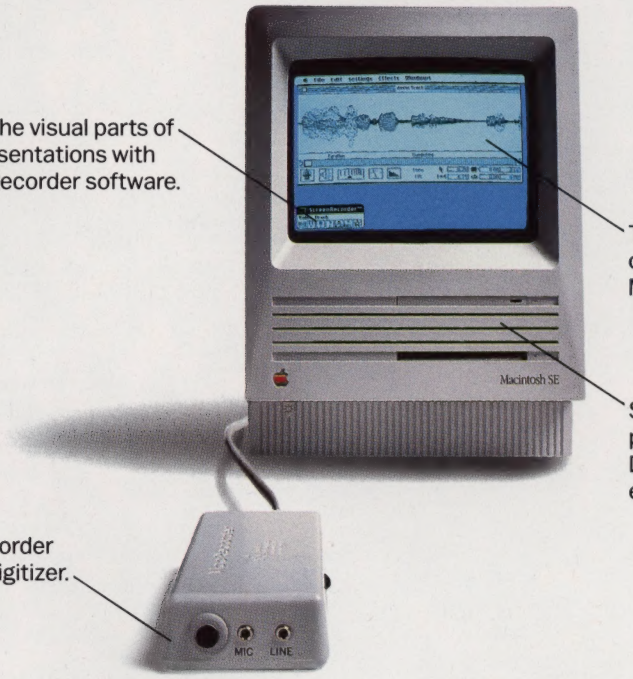
A Review of Culture 1.0: A HyperCard Humanities Course

Using ProViz Video Digitizers With HyperCard



05
0 74470 74465 8

Now see hear.



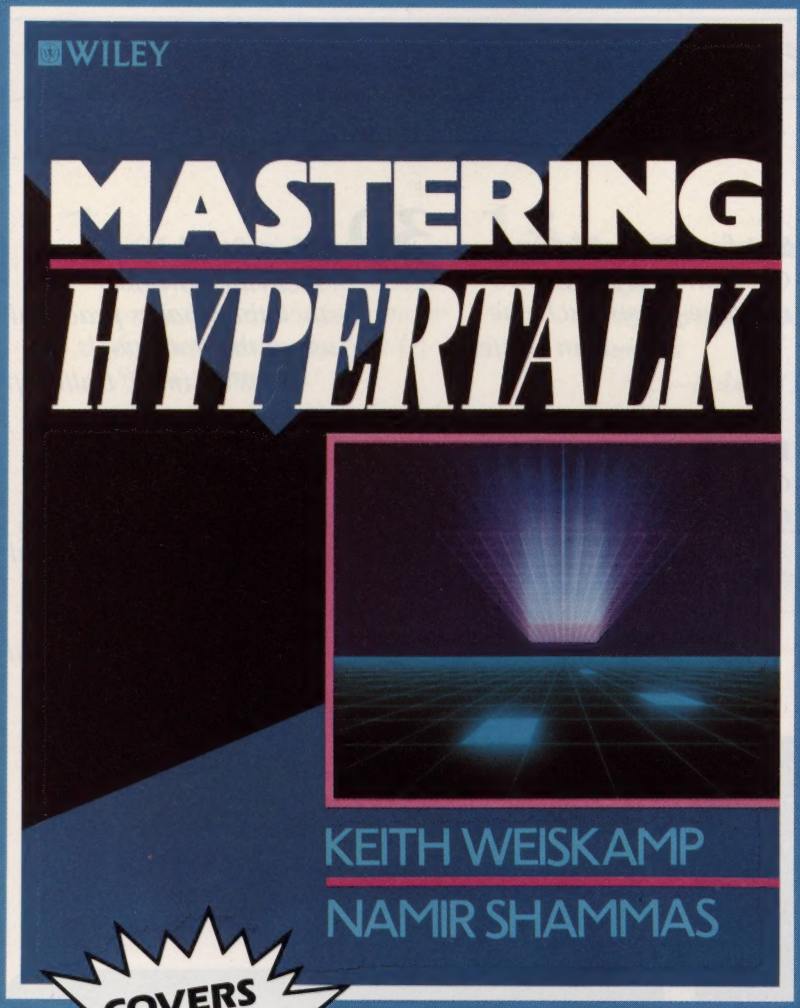
MacRecorder and ScreenRecorder will change the way you make, see, and hear presentations. Now it's easier than ever to create multimedia presentations right on your own Macintosh. You can watch the presentations on your screen, on someone else's screen—or on both at once. See and hear more about these new Farallon products at a dealer near you. Call (415) 849-2331. Stay tuned.

Farallon™

2201 Dwight Way, Berkeley, CA 94704

Fax # (415) 841-5770. MacRecorder and ScreenRecorder are trademarks of Farallon Computing, Inc.
Macintosh is a registered trademark of Apple Computer, Inc. © 1988 Farallon Computing, Inc.

maximum input = maximum output



To get the most out of HyperCard, you need to harness the full power of HyperTalk. Mastering HyperTalk gives you all the tips and techniques you'll need.

Major features include:

- A detailed appendix that covers the complete HyperTalk language version 1.2.
- Numerous examples of complete HyperTalk scripts.
- Discussions of important concepts such as object-oriented programming, hypertext, and techniques for developing scripting tools.
- Practical programming topics and algorithms include sorting, searching, graphics programming, and list and string processing.
- Stack applications that you can use or modify to suit your needs.
- Hands-on techniques for adding extensions to HyperTalk.

Get your copy today and get all the power of HyperTalk! $24.95

ABOUT THE AUTHORS: Keith Weiskamp is co-founder and editor of PC AI: The Artificial Intelligence Journal of Personal Computing and the author of Artificial Intelligence Progamming with Turbo Prolog. Namir Shammas, editor of Turbo Tech Report, has co-authored several computer books and written articles for BYTE, Journal of Pascal, Turbo Technix and other leading magazines in the field.

COVERS VERSION 1.2!

At bookstores or use this coupon to order today.

WILEY
605 Third Avenue
New York, NY 10158-0012

For fastest service, call
1-800-526-5368

ORDER NOW!

JOHN WILEY & SONS, INC. Attn: M. Schustack
605 Third Avenue, New York, NY 10158-0012

Please send me _____ copy(ies) of MASTERING HYPERTALK (1-61593-5) @ $24.95 per copy plus applicable sales tax.

☐ Payment enclosed, Wiley pays postage/handling.
☐ Bill Me ☐ Bill my firm/institution
Bill my ☐ VISA ☐ MasterCard ☐ American Express

Acct. # _____ Exp. _____

Signature _____

Name _____

Company _____

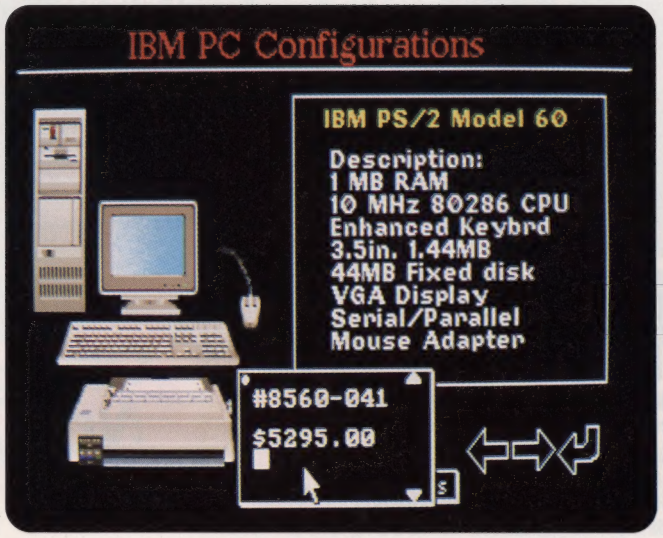
Address _____

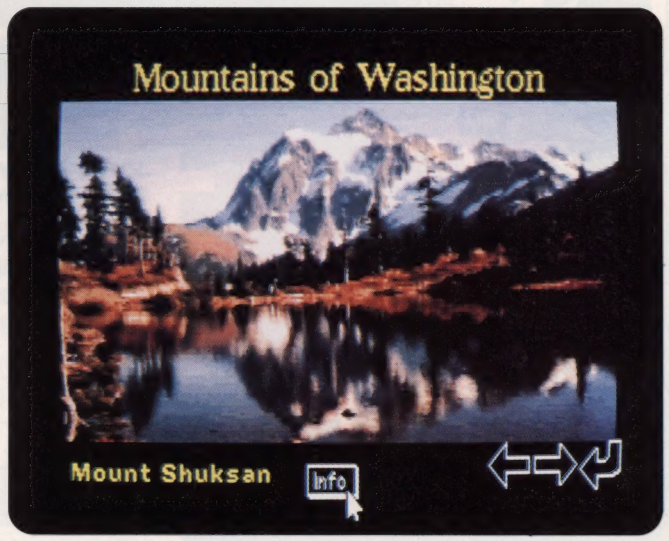
City/State/Zip _____

Price subject to change and higher in Canada.

HYPERLINK MAGAZINE

COLUMNS





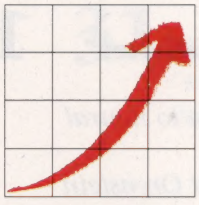
FEATURES

REVIEWS

Find out what's up at MACWORLD Expo.

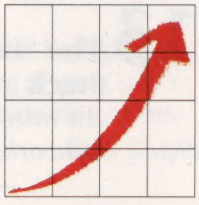
Attendance 1985 - 1988

San Francisco
Up 144%



Boston
Up 166%

Expo Booths 1985 - 1988



San Francisco
Up 300%

Boston
Up 280%

You can always count on seeing the newest, the best and the most at MACWORLD Expo. That's why, year after year, the original Macintosh-exclusive computer show keeps getting more and more popular, both for users and for vendors.

Fact is, MACWORLD Expo will:

• **Save you time and money . . .** by literally putting at your fingertips the hardware, software and peripherals that represent the entire state of the art of Macintosh™ computing. *You'll make smarter buying decisions.*

• **Build your knowledge . . .** by enabling you to attend a helpful tutorial that meets your specific need, whether you use your Mac at the office, at school or at home. *You'll learn from the experts.*

• **Develop your skills . . .** by giving you plenty of opportunities to practice what you've learned, using one of the many Macintoshes that will be available to you. *You'll get hands-on experience.*

• **Introduce you to the latest Macintosh products . . .** MACWORLD Expo has traditionally been *the* platform for major new product announcements such as the SE/30 and HyperCard. *You'll be at the forefront of new technology.*

• **Connect you with the best minds in the business . . .** through user group exchanges, advanced user tips sessions, and discussions with the industry's veteran techies. *You'll join a network of Macintosh pros.*

The next MACWORLD Expo to come in 1989 is:

BOSTON
August 10-12, 1989
Bayside Expo Center • World Trade Center

The next move is up. Just fill in, detach and return the coupon below to MACWORLD Expo, Box 155, Westwood, MA 02090. We'll mail you the information you need to: get a special reduced rate on your admission, avoid the registration lines and make the most of your time learning what's up.

What's up, MAC?

Please send me details about MACWORLD Expo/Boston 1989

❑ Exhibiting ❑ Attending

Name _____ Company _____

Street _____ City _____

State _____ Zip _____ Phone _____ HYP

Sponsored by MACWORLD, the Macintosh™ Magazine. An IDG Communications publication. MACWORLD Expo is an independent trade show not affiliated with Apple Computer, Inc. MAC, MACINTOSH and MACWORLD are trademarks of Apple Computer, Inc.

MACWORLD EXPOSITION

HYPERLINK MAGAZINE

May/June 1989 • Volume 2 No. 3

Publisher
David G. Brader

Editor
Roger Wood

Contributing Editors
Larry Burtness
James Paul
Robert Orenstein
Craig Ragland
Dan Shafer

Director of Design
Marv Boggs

Director of R & D
William K. Balthrop

Controller
Mark Andersen

Public Relations
Ken Jackson

Advertising Sales
Carole Eversole
(503) 484-5157

HyperLink Magazine is published 6 times a year by Publishers Guild, Inc., P.O. Box 7723, Eugene, OR 97401. Telephone (503) 484-5157.

Editorial material should be submitted to HyperLink Magazine, P.O. Box 7723, Eugene, OR 97401. Submissions will not be returned.

Publisher's Card

If only I had a Universal Button. *HyperCard* buttons are wonderful. You can train them to do all kinds of tricks–but only within *HyperCard*. If only I could create a *HyperCard* button from within any environment, be it the system desktop, *Microsoft Word*, *Excel*, or my General Ledger package. If only I could train that button to do anything at any level, to any environment, and be active in only the environments I choose.

It would work something like this:

Say I pull down a Mac Menu and select an imaginary "Universal Button..." option. A dialog box appears asking general questions such as, "Where shall this button be active?" with a scrolling window of environmental options to select. Then a familiar dialog box appears for specifying a *HyperCard* button. When we select the script editor from this box, an enhanced version of HyperTalk is available that includes commands for manipulation of the Mac's new operating system, system resources, and control of independent applications.

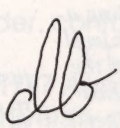
The new Universal Button(s) will have to exist inside a transparent Universal Shell Window that overlays everything else within the Mac (or IBM, or Sun, etc.).

Let's go a step further and say this Universal Button can have *HyperCard* field attributes or other object's attributes. Now we have a Universal Object. An extensible Universal Object with full access to any resource in the computer. Developing future applications depends on top programmers with a grasp of this environment and the existence of a very efficient HyperTalk compiler that can compile and then link modules of program code within a Universal Object's script. A standard set of pre-compiled "Shell Objects" will be supplied.

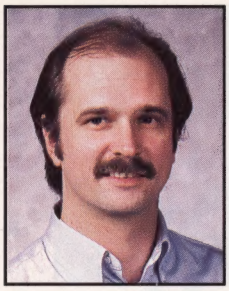
Is this the Object Oriented Interface of the near future? I think so. What things are pointing the way today? *HyperCard*, *SuperCard*, *IBM LinkWay*, and *Wingz* (this new graphic spreadsheet has buttons!) all have programmable objects and more such software is on the way. The desire for increased unification and standardization of the human/computer interface is driving the industry to a new standard. *HyperLink Magazine's* coverage of these topics will have some small influence.

Last issue we introduced Silicon Beach Software's *SuperCard* to you. This issue we introduce *IBM LinkWay*. In the future, we will show you how to build buttons in *Wingz*. We will continue to bring you useful solutions to your world of business, education, and personal productivity. To unlock your creative self through showing you Object Oriented Interface manipulation and the expanding control of your computing environment.

The future can wait while we explore today's Object Oriented Interfaces within the pages of this issue. Enjoy then, this latest edition of the world of *HyperLink*.



David G. Brader



Roger Wood
Editor

Why "set userLevel"

Have just received my Jan/Feb 1989 issue with its *Stack-Solutions* disk. This is, without doubt, the most useful issue yet. The direction of the magazine justifies the faith from the beginning.

I do have an aggravation. In Rhett Savage's excellent column, "HyperCard Haiku," there is a short script (Listing 10) which blocks `userLevel` changes. One of the great strengths of *HyperCard* as a medium is that we who are not professional programmers can learn techniques and solutions from problems we face by examining and adapting scripts of others. We do not have to reinvent the wheel. If that script starts showing up in general interest stacks, we will all be poorer.

Mr. Savage hints at a reversal without being specific. To get around that script, even at userLevel 1, send the following in the Message box:

```
send ¬
"set userLevel to 5" ¬
to HyperCard
```

As a matter of policy, I believe you should include reversing scripts for virtually anything you publish that attempts to block access to the medium.

Mark Merritt
Carmichael, CA 95608

Thanks for the comments Mark. Admittedly, impeding a user from setting the user level can be annoying, but it is the stack developer's prerogative to do so. Your method of getting around the script shows that to the crafty scripter, no barrier is too great.

Tips Feedback

In the process of implementing the script in the "HyperCard Tips" column for locking and unlocking a field in the Jan/Feb 1989 issue, I discovered an intriguing feature of *HyperCard*. When you hold down the Command key while clicking in a field, the text at the click is surrounded by a box and put into the Message box as well. Not only that, you can select several words, or a whole line, by dragging across the line while the Command key is down. Well, that led me to wonder if you couldn't use that as a handy way to implement a sort of Hypertext indexing technique. The script in Listing 1 is the result.

Of course, you can put any kind of searching script you like inside the inner `if` loop. The outer loop is not really necessary, unless you wish for the `mouseUp` handler to do something different if the field is locked and the Command key is not down.

A complaint: The magazine is for the most part quite useful. My complaint is that the experienced user is sometimes lost in the shuffle. A case in point: In the "Xpanding HyperCard" column explaining glue routines and calls to the *HyperCard* Toolbox, a passing mention was made of the passFlag variable. The function of that variable is not made entirely clear. Does *HyperCard* pass the XCMD name up the hierarchy before, during, or after the XCMD is run? What effect does this have on the operation of the XCMD if any? Is this just a ploy to sell books?

Gregory Krall
Albuquerque, NM 87015

The passFlag variable acts just like the `pass` *command in any handler that you write in HyperTalk. After the XCMD is done, if the passFlag variable is true then it passes the command (whatever the name of the XCMD is), up the object hierarchy. An XCMD is really not very different from any other handler, and if its name is the same as an already defined command in HyperTalk, setting the passFlag to true ensures that the normal HyperTalk command will execute. On the other hand, if the XCMD is designed to stop the normal action of a HyperTalk command, then such an XCMD would set the passFlag to false.*

Listing 1

```
on mouseUp
  put the commandKey is down into ¬
  commandKeyDown
  if commandKeyDown then
    get the message
    put it into commandSelection
    put empty into message
    hide message
    if  commandSelection is not empty then
      --put any searching script here
      --you desire
      push card
      go stack "A Lot of Text"
      find whole commandSelection in ¬
      background field "Index"
      If the result is empty then
        put the foundChunk into thisChunk
        select thisChunk
      else pop card
    else set the lockText of me to not ¬
    the lockText of me
  end if
end mouseUp
```

Invisible Objects

Enclosed is the information for a stack that demonstrates a strange quirk in the HyperCard environment. As you know, the rect of a given button or field is given by top-left x, top-left y, bottom-right x, bottom-right y. Since this is the format, it is assumed that the top-left will be further up and to the left of the bottom-right point. But if this is not the case, the button or field is "turned inside out," and is not visible even if the visible of the object is true. This is a neat trick for showing and hiding buttons and fields, because pressing Command-Option-Shift will not show these objects (of course this could be dangerous also!!!)

Brad Smith
Kismet Technologies
Laurel, MD 20708

Very interesting, Brad—and true too! If you type

```
set the rect of button 1
to 100, 100, 50, 50
```

into the Message box (no line break), or put a similar command into a script, when the command is executed, button 1 disappears. If you then type

```
put the loc of button 1
```

into the Message box, HyperCard *responds with*

```
75,75
```

in the Message box—equally strange. An interesting anomaly—anyone have any ideas as to how this could be a useful technique?

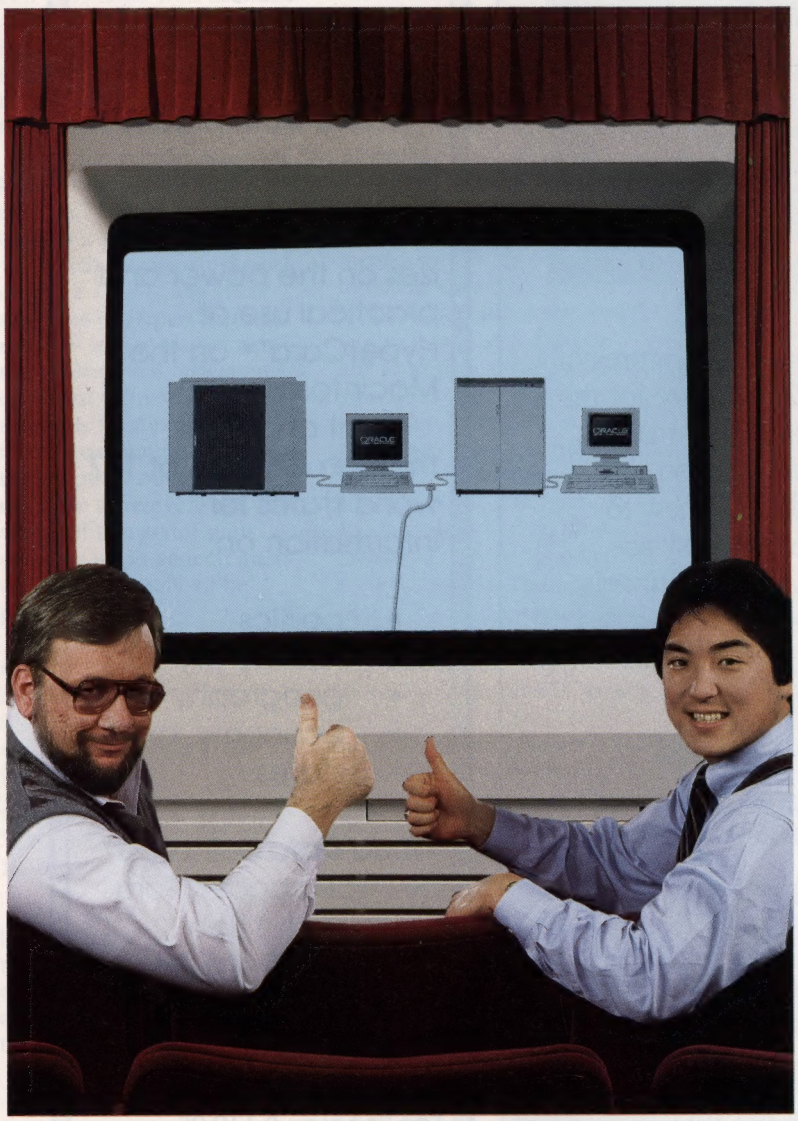
SuperCard Update

Silicon Beach Software, devlopers of *SuperCard*, (see the "Special Report" in the March/April 1989 issue of *HyperLink)* told us as we went to press that *SuperCard* is still on schedule for a second quarter ship date. When pressed, they said it would be released "as soon after MacWorld Washington as possible." (MacWorld Expo Washington is to be held on April 25-28.)

In a related matter, at the product announcement of *SuperCard* in San Francisco in January, Apple Computer's Jean-Louis Gassée referred to a "HyperTalk Language Standards Commitee." At least a preliminary meeting of this committee has been held. Among the attendees were Mike Holm, head of Technical Assistance on Apple's *HyperCard* team, and representatives from Silicon Beach Software.

"Two thumbs up!" ORACLE for Macintosh. For only $199 until May 31.*

LIMITED TIME OFFER: Purchase ORACLE for Macintosh Networking Version and receive ORACLE for 4th DIMENSION **FREE!**



Finally, there's one thing these two guys can agree on: ORACLE® for Macintosh. That's because ORACLE for Macintosh now turns both HyperCard and 4th DIMENSION into full-function SQL databases. It also gives them transparent connectivity to over 80 different systems, including PCs, DEC VAXs and IBM mainframes—even IBM DB2 and SQL/DS databases.

No-Risk 30-Day Money-Back Guarantee

Whatever application you're currently using—HyperCard or 4th DIMENSION—ORACLE for Macintosh gives you industry-standard SQL. $199* delivers the first and only SQL database available for the Mac. An additional $199 lets you access ORACLE from 4th DIMENSION. Or for a limited time, $999** delivers all this, plus transparent access to remote ORACLE and IBM databases on your host systems. So call today. Our 30-day money-back guarantee is your assurance we'll deliver a four-star performance.

ORACLE®

COMPATIBILITY · PORTABILITY · CONNECTABILITY

Call 1-800-ORACLE1, ext. 9235 today.

"I like the seamless way ORACLE gives HyperCard full-power database and networking capabilities. Teaming the total flexibility of HyperCard with the strength of SQL is a masterstroke!"

Dan Shafer, *Author of Hypertalk Programming*

"I like the seamless way ORACLE opens up minis and mainframes to 4th DIMENSION. Teaming the total flexibility and strength of 4th DIMENSION with SQL is a masterstroke, indeed."

Guy Kawasaki, *President of ACIUS, developers of 4th DIMENSION*

Dear Oracle Macintosh Direct Sales
20 Davis Dr. • Belmont, CA 94002 • **1-800-ORACLE1, ext. 9235**

I have *(check one)* ☐ HyperCard 1.2 ☐ 4th DIMENSION Version 1.0.6.
Enclosed is my ☐ check, or ☐ VISA ☐ MasterCard ☐ AmEx credit card authorization for:

___ × $199 ORACLE for Macintosh Developer's Version $ _____
(Includes HyperCard Interface. $299 after May 31, 1989.)

___ × $199 ORACLE for 4th DIMENSION $ _____
Available June 1989. (Requires ORACLE for Macintosh and 4th DIMENSION.)

___ × $999 ORACLE for Macintosh Networking Version $ _____
(Includes an unrestricted license of ORACLE for Macintosh, SQL•Net Networking Software and protocols for Macintosh and, for a limited time, ORACLE for 4th DIMENSION.)

Please add appropriate sales tax $ _____

Shipping and handling $ 15

Total *(Offer valid only in USA)* $ _____

NAME TITLE

COMPANY

STREET *(no P.O. boxes, please)*

CITY STATE ZIP

TELEPHONE

CREDIT CARD NO. CARD EXPIRATION DATE

SIGNATURE TODAY'S DATE **HYPERLINK**

*Price becomes $299 after May 31, 1989. Stand-alone version licensed for developers only. Requires Macintosh SE or II with 2 MB of RAM, 5 MB hard disk space, floppy disk drive, and HyperCard 1.2. Includes 30-day installation support, ORACLE database, Hyper•SQL (HyperCard Interface), SQL•Plus, OCI and Pro•C (Macintosh Programmer's Workshop required for programming usage), System Stacks and Example Stacks. **Full networking version is $999 and includes SQL•Net (for database communications), Async, DECNet, TCP/IP support, Alisa System's TSSnet DECnet protocol and drivers and Kinetic's TCPort TCP/IP protocol and driver. Accessing database software on other machines requires a separate protocol handler and gateway software for the other machine. Call for additional information. ORACLE is a registered trademark of Oracle. The other companies mentioned own numerous trademarks. TRBA

Here Are Some How-to Tips on Using Some of the More Frequently Encountered SQL Commands

SQL Basics, Part 2

by Dan Shafer

Editor's Note: This is the final article in a series of articles by HyperTalk author, columnist, and consultant Dan Shafer on the design and programming of *HyperCard* front ends for databases. Here, Shafer concludes by examining several commands in Structured Query Language (SQL)— the language used by industry-standard mainframe and minicomputer databases. This information is useful to anyone building a *HyperCard* front end for such databases, whether one uses *ORACLE for Macintosh* or another approach, because SQL has become a *de facto* industry standard for database access.

In the last issue, I introduced the basic SQL commands for setting up a database for a delicatessen. We concentrated in that discussion on the `CREATE TABLE` and `INSERT` commands that are used to build database tables and place values into them. In this issue, we'll turn our attention to the process of extracting information from SQL databases with the deceptively simple `EXTRACT` command.

This application consists of seven tables:

- INGRED (containing ingredients used by the deli)
- CUST (containing customer information)
- ADDRESSES (containing building addresses of major clients with multiple locations)
- CUST_TYPES (containing customer type codes and their descriptions)
- ORDERS (containing ordering and payment information)
- ORD_TYPE (containing order type codes and their descriptions)

The CUST table has the structure shown in Figure 1. Figure 2 shows the contents of the file that we will assume in our discussion of the `SELECT` command.

Column Name	Type	Size
Customer ID	Character	5
Last Name	Character	24
First Name	Character	24
Company	Character	24
Building	Character	4
Phone	Character	8
Extension	Character	3
Customer Type	Character	3

Figure 1
Structure of CUST Table

Starting Simply

The simplest form of the `SELECT` command tells SQL to extract all of the contents of all of the columns from a named database table. To extract all of the data from the CUST table in our example, the command is:

```
SELECT * FROM CUST;
```

Used without *HyperCard* on a non-graphic terminal (or under MPW on the Macintosh), this would produce a tabular listing of all of the contents of all of the fields, much like that shown in Figure 2, with column headings. The asterisk simply tells SQL to select all of the columns in the table.

We often want to choose information from only selected columns, however. In that event, we just replace the asterisk in our previous example with a comma-delimited list of the fields we wish to extract. For example, let's say we want to build a phone list we can print and keep next to the phone for returning calls and making sales contacts. We want this listing to show us the customer's last name, first name, company, phone, and extension. Here's the SQL command that would produce such a listing:

```
SELECT LAST_NAME, FIRST_NAME, COMPANY, PHONE,
EXTENSION
FROM CUST;
```

Again, assuming we are running the SQL database outside *HyperCard*, the result of this command would be a neat, if somewhat plain-looking, tabular listing of all of the customers in our database, showing each one's name, company, and phone contact information. In fact, all SQL `SELECT` commands produce such listings. When we talk about how this command is used with some extensions in Hyper*SQL from *ORACLE* in conjunction

Dan Shafer is the author of HyperLink's *regular "Shafer On Scripting" column. He is also President of Strategy Consulting, the first certified* ORACLE *Mac consultant, and author of the soon-to-be-published book* Using ORACLE with HyperCard *(Hayden Books).*

with a *HyperCard* stack, we'll see the other presentation options from which we can choose.

Sorting in SQL

The two previous example commands produce tabular reports on the display and present the data in the order in which it is stored in the database. This order probably bears no relationship to reality even though we've shown in Figure 2 that the data is stored alphabetically by customer last name. It is probably stored somewhat randomly, particularly if additions and deletions to the data have been made since the table was first loaded. If we wanted the customer list sorted by customer last name, for example, we could write an SQL query like this:

```
SELECT * FROM CUST
ORDER BY LAST_NAME;
```

Similarly, if we wanted to put the phone list in company order, we could write a query like this one:

```
SELECT LAST_NAME, FIRST_NAME,
COMPANY, PHONE, EXTENSION
FROM CUST
ORDER BY COMPANY;
```

As you can see, SQL's sorting capability is quite straightforward. You can specify sorts in descending order by placing `DESC` after the word `COMPANY` above.

Only Want to See...

We rarely want to see all of the data in a file, or database table. We have already seen how to describe the columns to be extracted from the table. But what if we want only those records (rows in SQL parlance) that match certain selection criteria? In SQL, this kind of selective data extraction is accomplished with the `WHERE` clause in a `SELECT` command. For example, let's say we want a phone list that only shows our regular customers (i.e., those with a Customer Type of `REG` in Figure 2). The SQL command selecting the appropriate columns from only those rows would be:

```
SELECT LAST_NAME, FIRST_NAME,
COMPANY, PHONE, EXTENSION
FROM CUST
WHERE CUSTOMER_TYPE = 'REG'
ORDER BY COMPANY;
```

Cust ID	Last Name	First Name	Company	Bldg.	Phone	Ext.	Cust. Type
ALLN1	Allen	Jerry	National Can	43	123-4567	222	REG
BACN1	Bacon	Sharon	National Can	42	123-5678	123	CONF
ELSN1	Ellison	Laurie	Delphi Corp		555-4123		REG
HOWD1	Howard	Stephanie	National Can	42	123-5678	768	REG
HOWD2	Howard	Andrew	National Can	42	123-5678	769	OCC
JAMS1	James	Burton	Delphi Corp		555-4223		OCC
MILR3	Miller	Andrea	Smythe Incline		487-0009	123	CONF
MILR5	Miller	Aaron	Delphi Corp		555-4422		OCC
SCHF1	Schafer	Christine	Apricot Press		544-8540		OCC
WLSN3	Wilson	Ferdinand	National Can	44	123-4565	214	CONF

Figure 2
Typical contents of a file stored in the structure of CUST Table shown in Figure 1

Operator	Meaning
=	equal to
<>, ^=, or !=	not equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
BETWEEN...AND...	between two values
IN	contained in a list provided as an argument
LIKE	matching a pattern of characters
IS NULL	value is null (i.e., has no value)

Table 1
SQL Logical Operators

You now begin to get some of the "flavor" of SQL. Much of your work with SQL will consist of creating `SELECT` commands of varying degrees of complexity. And these commands can become extremely complex. For example, let's say we now want to limit the phone list to regular customers who work for National Can. The query would look like this:

```
SELECT LAST_NAME, FIRST_NAME,
COMPANY, PHONE, EXTENSION
FROM CUST
WHERE CUSTOMER_TYPE = 'REG'
AND COMPANY = 'National Can'
ORDER BY COMPANY;
```

Notice the word `AND` joining two conditions: `CUSTOMER_TYPE = 'REG'` and the newly added `COMPANY = 'National Can'` entry. This would produce a phone list containing only the names and contact data for Jerry Allen and Stephanie Howard from our example in Figure 2.

This ability to perform selective or conditional data extraction from SQL tables is among the language's most powerful features. Table 1 shows the various logical operators that can be used in SQL queries. As you can see, the range is far more extensive than that available in HyperTalk or in any general-purpose programming language.

As one example of the power of these logical operators, consider the possibility that we want to print a list of the names and companies of all of our customers who fall into any category other than occasional (i.e., with a `CUSTOMER_TYPE` other than `OCC`). The following SQL query would do the trick:

```
SELECT LAST_NAME, FIRST_NAME,
COMPANY
FROM CUST
WHERE CUST_TYPE 'OCC'  =
CUST_TYPES.CUST_TYPE
ORDER BY COMPANY;
```

As you may already have anticipated, we can also use an `OR` logical connector as we have used the `AND` in previous examples. To get a list of all of our customers who are either regular customers or who work at Apricot Press, for example, we could write the following query:

—continued on page 24

An Introduction to Neural Networks

The "Letter Learner" Stack

by Robert Orenstein

Neural networks are a hot topic in computing right now. The general computer user's interest in neural networks is exceeded only by his inability to understand most of the articles that have recently been written about them. This is the fault of the literature, not of the reader. Most of the articles are of a highly technical nature, and many describe methods of using neural networks to accomplish tasks that could easily have been accomplished using ordinary simple programs. For example, an article in another magazine described how to use a neural network to count votes for one of two candidates in order to determine which candidate won an election. Clearly a neural network is not necessary to accomplish this task.

"Letter Learner" is a HyperCard *stack, and its safe to say that without* HyperCard, *I'd still be just starting the project.*

I've recently written a stack, called NeuralStack One: "Letter Learner" that is meant to explain neural networks in an intuitive, easy-to-understand way. Figure 1 is thte basic screen of "Letter Learner." In this article, I will first discuss neural networks in general, and then move to a specific discussion of "Letter Learner." Weighing in at a hefty 400,000 bytes, including 6 *Turbo Pascal* XCMD's, "Letter Learner" is not a stack that you can easily replicate on your own. However, in this article, I hope to give you enough information to understand the basic ideas behind neural networks, and to understand how "Letter Learner" works.

What Is a Neural Network?

If you were to compare a twenty-page description of a Macintosh with a twenty-page description of an IBM mainframe, you would find the two machines to be very different. If you were to read five-page descriptions of the same machines, the descriptions would be more similar. A five-*sentence* description of the two machines would be identical to each other. That's because fundamentally, all computers today use the same basic architecture (something called the Von Neumann model), and the differences between two machines are simply differences in the way that the basic model is implemented. If I were to give you a five-sentence description of a neural network, however, it would be entirely dif-

Robert Orenstein is a HyperCard *consultant in Venice, California. He is currently working on NeuralStacks Two and Three.*

"Letter Learner" Stack Availability

NeuralStack One: "Letter Learner" is a shareware stack that is currently available from three sources. It is included on this issue's *HyperLink StackSolutions* disk (see the inside back cover of this issue for order information). The Berkeley Macintosh Users Group has the program available as well, and other users groups will soon be distributing it. It is also available directly from the author at P.O. Box 1381, Venice, CA 90294. If you wish to get a copy directly, please send the $10.00 shareware fee to the above address. In order to receive NeuralStacks Two and Three, the author requires payment of the shareware fee for "Letter Learner."

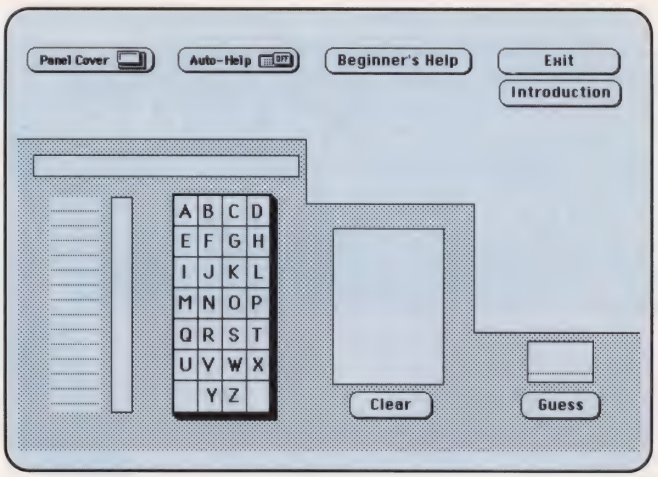


Figure 1

When you first start Letter Learner, it looks like this. Through drawing letters with the mouse in the box in lower middle of the screen, and pressing on the appropriate letter in the box to the left, you "teach" the stack what the letter is by example.

ferent. That's because a neural network is unlike any machine currently available. Rather than use the Von Neumann model, a neural network takes the basic concepts underlying the brain as its architecture. In fact, the five-sentence description of the brain and the five-sentence description of a neural network would be identical, even if the twenty-page descriptions are entirely different. So what are some of the differences between the standard computer and a neural network?

Neural Networks Are Not Programmed

Any standard computer must be programmed. If I wanted one to learn the differences between twenty peoples faces, I would have to program the machine with a list of rules to tell the faces apart. Neural networks are not programmed; rather, they are trained. If I want a neural network to tell twenty people apart, all I have to do is show it a few examples of each face.

This is the same way that people learn. When you were a baby, you learned your parents' faces without any rules explained to you to help you do so. How did you do it? You learned *by example.* You saw your parents' faces 500 times or so, and eventually you could tell one from the other.

Neural networks work in a similar way, and use the same basic methods that the brain does. Both the brain and neural networks are made up of neurons, which are connected by synapses. Knowledge is stored in the synapses, by the strength of the connections between the neurons. (In a neural network, neurons are usually given other names, such as "processing elements" or "units." I have chosen the term "neurode," which is more descriptive, and keeps the analogy to neurons evident). Although neural networks are similar to brains, the brain is *billions* of times more complex, and although the brain can adapt itself to many tasks, a particular neural network is usually built to perform only one task.

If a neural network is different from a regular computer, then how can we run a neural network on a Macintosh? Rather than build the neural network from hardware, we can *simulate* it on the Mac via a program. The program contains instructions that allow it to learn from experience. It runs more slowly than a true hardware neural network. For example, even if we were to program a neural network on a Cray XMP, the fastest machine currently built, we would have less power than we need to simulate a leech.

Figure 2

Uses of Neural Networks

So what can we use neural networks for? Although they have many functions, the class of problems that I find the most interesting involve pattern classification. These problems involve looking at a particular object and deciding which category the object falls in. For example, which letter of the alphabet has been drawn in Figure 2?

Trivial task, right? For you, yes. For a computer, no. Think about it—you've never seen that particular letter R before, exactly that shape and size, and yet you could immediately classify it as an R. And, unlike an ordinary computer program, you don't have to learn any rules to be able to do this. You simply have accumulated experience in recognizing letters, and this allows you to look at this particular pattern of lines and classify it into one of 26 categories, the letters of the alphabet.

Neural networks are built to handle exactly this type of task. A voice recognition neural network might be able to listen to a voice and tell you whose voice it is. A neural network currently in use at a bank acts as a loan officer—based on past examples, it can look at a loan application and classify it as "Good loan risk" or "Bad loan risk."

—continued on page 43

This Depreciation Calculator Can Save You Time and Money

The "Fixed Assets" Stack

by William K. Balthrop

Housing and equipping a business can require a large investment in property. The property may be land, buildings, office furniture, tools, even copyrights and patents. This property is referred to as fixed or long-lived assets.

There are two principal types of fixed assets: tangible and intangible. Tangible assets are classified as either real property and personal property. Real property includes things such as land, buildings, or other structures attached to the land. Personal property consists of items such as office furniture, equipment, tools, and fixtures. The intangible assets of a company include copyrights, trademarks, patents, organization costs, and goodwill.

The first thing you need to know about an asset is the cost of acquisition. This includes the purchase price, shipping charges, and any expense such as installation charges, required to make the asset ready for service.

Through the use of an asset a company accrues normal expenses such as maintenance, repairs, and insurance. All of these costs are visible and deductible. There is also an invisible cost which must be accounted for. As most assets are used by the company they become old and wear out. An asset has only a limited life before it must be retired or salvaged. There are three major terms used by accountants to account for this expense: depreciation, depletion, and amortization.

Depreciation is the term used to describe the transfer of an asset's acquisition cost to an expense which can be deducted. Assets which can be depreciated are buildings, machinery, equipment, and fixtures. Land can not be depreciated, as land once used, is always still land and does not wear out. Things which grow on the land, however, such as timber, or reside within the land such as minerals or oil, however can be removed. Removing these items does decrease the value of the land, and Depletion is the term used to describe this.

Just as tangible assets get old and wear out from use, many intangible assets also have a useful and legal life span. The acquisition cost of these assets should be spread over their life span. This process is called amortization.

Before this article turns into an accounting text book, let's just say that there is a lot of work involved in tracking all of the assets within even a small company. We have developed a *HyperCard* stack that takes much of the drudgery out of tracking assets. The "Fixed Assets" stack lets you to maintain a database of your assets and helps you figure the depreciation schedules for those assets.

Editor's note: Because of the stack's large size and complexity, we couldn't list the entire stack in the magazine. We are including the stack script which contains the handlers for calculating the depreciation schedules. Using these as a guide, you can create your own stack to do the complex and tedious math calculations required. The entire "Fixed Assets" stack is, of course, included on this issue's *StackSolutions* disk.

Using the "Fixed Assets" Stack

Upon opening the stack you are presented with a screen like the one shown in Figure 1. Enter a description of the asset along with any address or location information. If your assets are currently filed in another manner with reference numbers, you may include your own reference number, or by clicking on the "Auto" but-

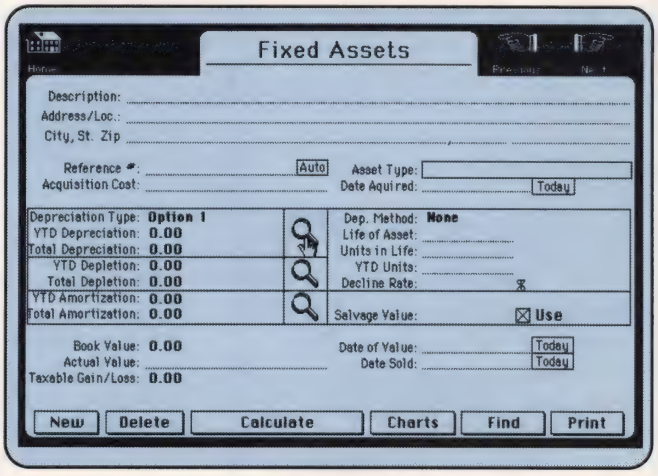


Figure 1

William "Kelly" Balthrop is the Director of R & D at HyperLink *and has been developing and publishing computer software for the last decade.*

ton, you can generate a new unique reference number for your asset. Below the reference number enter the total acquisition cost of the asset. Remember this includes all costs required to put the asset into service.

To the right of the reference number is the asset type. Click on the rectangle to the right of "Asset Type." This displays a list of types from which to choose. Click on any one to select it. If you would like to change an entry or add your own custom option, click on the lock icon in the upper-left corner of the field. This unlocks the text in the field so that it may be edited. To lock the field so you can make a selection, click on the lock symbol again. Just click outside the field to cancel the entry of a new option.

Below the asset type is the "Date Acquired" field. You may enter the day the asset was placed into service, or simply press the "Today" button to automatically enter the current date.

On the left side of the screen, below the acquisition cost is the "Depreciation Type" field. Each asset can simultaneously maintain up to three depreciation options using different methods of depreciation. This field indicates which of those three options is being used to calculate depreciation to date. To switch which option is being used, click on the current option. The three options are listed in a field. Simply select one by clicking on it.

After selecting it, the method being used by that option is displayed in the field "Depreciation Method" to the right. In Figures 2 and 3 we see that Option 1 is selected and is using Straight-Line depreciation. If the option selected were Manual, then the fields for Depletion and Amortization would unlock so you could enter whatever you want for YTD (Year to Date) and total values.

The "Life of Asset" field is the useful and legal life given to an asset in years. To determine the this for your assets you should consult your accountant or a tax guide. The number of years you enter determines how many years over which the depreciation is spread.

The "Units in Life" field is used only when you are going to use the

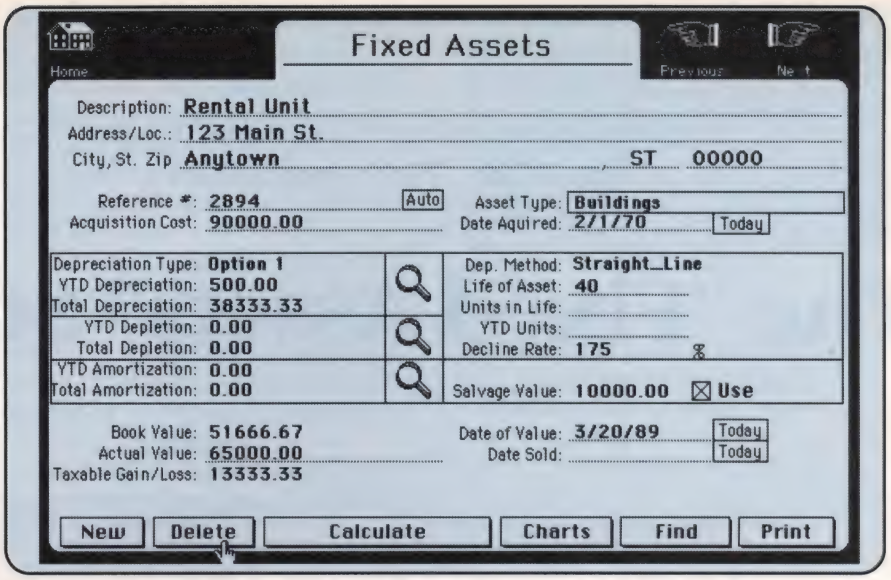


Figure 2

Units-of-Output method of depreciation. Lets say, for example, you buy a new stamping press. This press is rated to have a lifetime of one million stamp cycles. For every stamp cycle the machine is put through you will be able to depreciate 1 millionth of the acquisition cost. If you run the machine through 100,000 cycles in one year, you could depreciate 10% of the acquisition cost. In the "Units of Life" field, you should indicate the total number of units in the asse's life. In the case of our example that would be 1000000 (Please note: Do not use commas when entering large numbers into fields).

The "YTD Units" field is automatically updated from the table that is compiled for the Units-of-Output method of depreciation. If Units-of-Output is not being used this value should be 0 or left empty.

The "Decline Rate" field indicates upon what rate the declining balance is based, if the declining balance is the depreciation type being used. A 200% declining balance is also called double-declining balance, and is the maximum rate legally allowed. A 100% declining balance is calculated the same as straight-line depreciation, so the decline rate must be between these two values.

Below the declining balance rate is the "Salvage Value." When an asset wears out, it might be salvaged for some amount of money—e.g., as scrap metal or other recyclable material. This salvage value is what the asset would be worth after the life of the asset has expired. The salvage value may be deducted from the acquisition cost before doing the depreciation calculation. If you want the salvage value deducted from the acquisition cost, then enter the salvage cost, and click on the button to the right marked "Use." You may retain the salvage value for your records, and not have it deducted from the acquisition cost by not clicking on the "Use" button. When calculating the declining-balance method of depreciation, no salvage value is allowed. The stack takes care of this for you, even if you select the "Use" button, by not including the salvage value in declining-balance calculations.

The Tables

In the center of the screen in Figure 2 are three magnifying glasses. Clicking on one of these lets you view the tables of information for depreciation, depletion, and amortization. Clicking on the top magnifying glass shows you the three tables used for calculating depreciation (see Figure 3). The left side of each table shows the years that make up the asset's life. At the top of each column is a box. Click on this box to display a list of the depreciation options. The following options may be selected:

- None
- Straight-Line
- Declining-Balance
- Declining-Strait

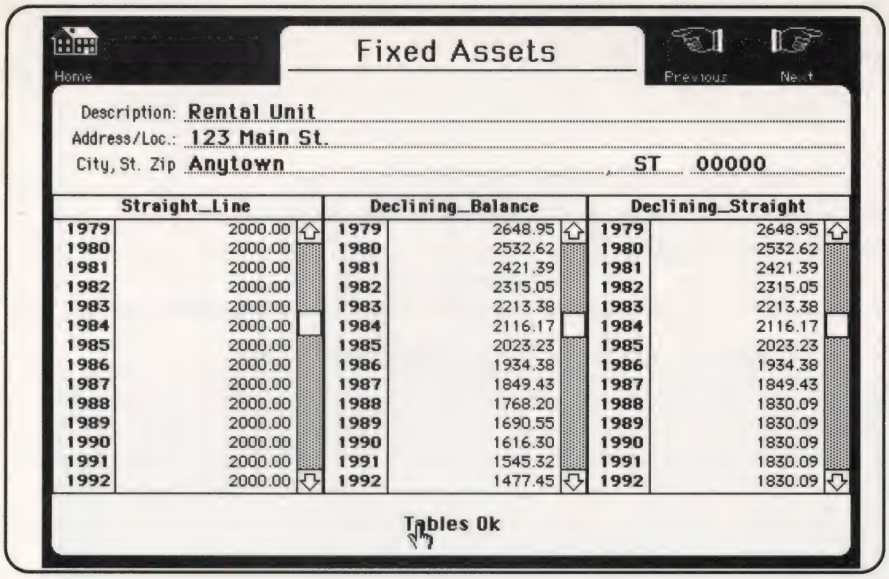


Figure 3

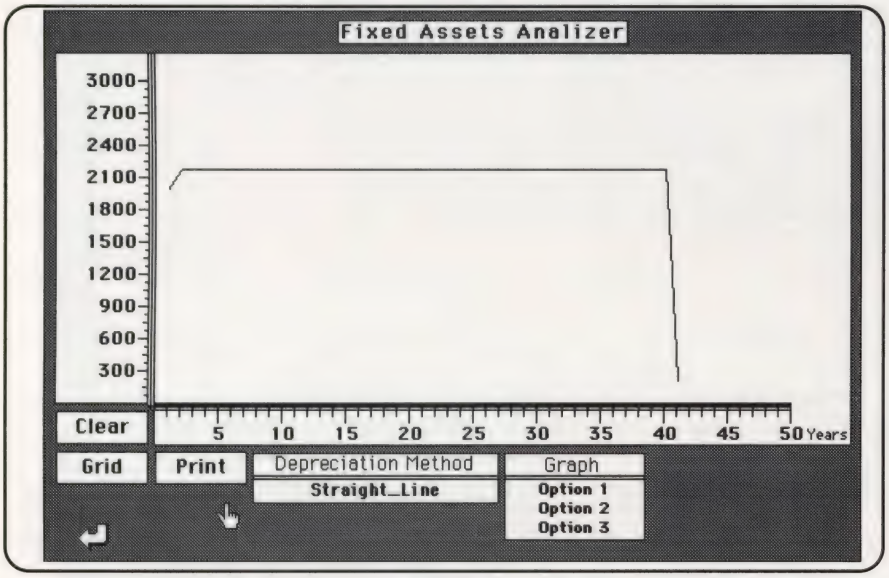


Figure 4

- Sum-of-Years-Digits
- Units-of-Output
- Manual

Click on one of these options to calculate the table. When it is done, the method is displayed in the box, and the depreciation for each year is placed in the field next to the years. The entire life of the asset is calculated, even if the asset is only one day old. You can enter up to three different methods of depreciation and view them side-by-side to see the differences between depreciation methods. You can later chart these values with a built-in charting screen (see Figure 4). In order to calculate year to date (YTD) depreciation on the main screen you need to have at least one of these options filled out with values for yearly depreciation.

If you are using the "Units-of-Output" option with the tables, you must enter the number of units followed by a comma, followed by the amount of money the items are worth.

To enter values for depletion click on the middle magnifying glass. A single table is displayed with a line for each year. Manually enter the depletion for each year, then click anywhere outside the field to close it. Clicking on the magnifying glass for amortization works in the same way as depletion.

Note that "Units-of-Output" and depletion are simply two ways of figuring the same thing. The only difference is that the amount that is entered into the "Depletion" field is in dollars, and the amount entered into the "Units-of-Output" table is both units and dollars separated by commas. The stack does not verify that the dollar amounts are identical, you need to check that they are the same if that is appropriate.

Depletion and amortization values are computed based on the values entered into each table. The YTD figure represents the depletion or amortization for the current year, and the total value is the accumulated total to date.

In the lower-right corner of the screen are two date fields. "Date of Value" is the date that is used to do all calculations. You may click on the button "Today" to have today's date automatically entered. The "Date Sold" field is for reference only. If you wish to calculate the Book value as of the date sold, enter the same date into both of fields.

The "Book Value" is calculated by subtracting total depreciation, depletion, or amortization from the acquisition cost of the asset. The "Actual Value" is either the price obtained through salvage, sale or simply an estimated value if the asset is till in service. The "Book Value" is then subtracted from the "Actual Value" to obtain the "Taxable Gain/Loss" for the asset. This value indicates your taxable equity in the asset.

There are several buttons at the bottom of the screen (see Figure 1 or 2) that help you maintain your database of assets. Table 1 shows a list of the buttons and their purposes.

Charts

When you select the "Charts" button at the bottom of the screen, you are taken to the "Charts" card. From here you can plot any of the three depreciation options which were set up on the main card. These depreciation tables must already be entered before you can plot them. The example shown was created by selecting "Option 2" from the "Graph" field at the bottom of the screen. Select an option simply by

New - Creates a new asset card.
Delete - Deletes the current card.
Calculate - Calculates the YTD and Total depreciation, depletion, and amortization. It also calculates the current book value and your taxable gain or loss.
Charts - Takes you to the "Charts" card (discussed below)
Find - Allows searching for assets by the description field.
Print - Prints the current screen.

Table 1

A list of the buttons across the bottom of the main screen of the "Fixed Assets" stack.

Listing 1 - Stack Script for "Fixed Assets" stack

```
on OpenStack
  global DelCard
  put false into DelCard
  hide menubar
  hide msg
end OpenStack

on doMenu Command
  global DelCard
  if DelCard is true then
    put false into DelCard
    pass doMenu
    exit doMenu
  end if
  if Command is "New Card"
  then send "mouseUp" to bg btn "New"
  else if Command is "Delete Card" then
    send "mouseUp" to bg btn "Delete"
  else pass doMenu
end doMenu

on None
  global Origin,Graph
  put empty into Graph
  if Origin is "Opt1" then put Graph into fld "Opt1 Values"
  if Origin is "Opt2" then put Graph into fld "Opt2 Values"
  if Origin is "Opt3" then put Graph into fld "Opt3 Values"
end None

on Straight_Line
 global AssetAge,Life,Book,Lmonth,Fmonth,Fm,Lm,Fyear
 global Lyear,RealAge,flag2,Per1,AqCost,Origin,Graph
  SetUp
  if not flag2 then exit Straight_Line
  put empty into Graph
  put 0 into Total
  put CheckNum(Book / Life * Fm) into line 1 of Graph
  add line 1 of Graph to Total
  get CheckNum(Book / Life)
  repeat with x=2 to trunc(Life) + 2
    set cursor to busy
    if Total + it > Book then
      put CheckNum(Book - Total) into line x of Graph
      exit repeat
    end if
    put CheckNum(Book / Life) into line x of Graph
    add it to Total
  end repeat
  if Origin is "Opt1" then put Graph into fld "Opt1 Values"
  if Origin is "Opt2" then put Graph into fld "Opt2 Values"
  if Origin is "Opt3" then put Graph into fld "Opt3 Values"
end Straight_Line

on Declining_Balance
 global AssetAge,Life,Book,Lmonth,Fmonth
```

OpenStack - Hides the menu bar when the stack is opened.

None - Clears the table when no depreciation schedule is desired.

Straight_Line - This routine is responsible for calculating straight line depreciation. It builds the table for the life of the asset, then plugs that table into the appropriate option table. It is the simplest type of depreciation.

Declining_Balance - This routine calculates the declining balance depreciation table. The resulting table is then plugged into the appropriate option table. To use this method you must have value between 100 and 200 entered into the "Declining Rate" field on the main card.

Declining_Straight - This routine calculates a declining balance depreciation until such a time that a straight-line depreciation allows for more depreciation than the declining balance method. The optimum point at which the routine switches from declining balance to straight line is automatically calculated.

Sum_of_Years_Digits - As the name implies, this routine calculates the Sum of Years Digits method of depreciation.

Units_of_Output - This routine calculates the units of output. In order for this routine to work, you must first enter the number of units used in each year into the table. Then select the Units_of_Output option. The money amount is put into the second item of each line, and the number of units is maintained in the first item.

Calculate - This routine is responsible for taking the selected table of depreciation values, and calculating the YTD depreciation for the date of value, as well as the total accumulated depreciation.

Manual - This is a dummy routine created simply to interrupt the system message being sent when the Manual method of depreciation is selected.

Finish - This routine calculates Book value and the taxable gain or loss for the asset.

FinishAmortization - This routine updates the YTD and total amortization values from the amortization table, based on the date of value.

FinishDepletion - This routine updates the YTD and total depletion values from the depletion table based on the date of value.

FinishUnits - This routine updates the "Units Used" field from the depreciation table that is also used to maintain the number of units used each year.

SetUp - This routine sets up many of the parameters required for calculating depreciation.

Age - This routine calculates an asset's age, as well as the fractional part of the first year of depreciation and the fractional part of the final year for calculating the current date depreciation.

Graphit - This routine uses one of the three depreciation tables created on the main card, and draws a line graph of the asset's depreciation.

ClearScreen - This routine clears the graph.

CheckNum - This commonly used function is responsible for validating legal numbers and converting them into the proper format.

ClickLine - This function returns the line clicked on within a field when called from within that field's script. There are two modes to this function:

- **ClickLine( )** - returns the line number clicked on.
- **ClickLine(stuff)** - returns the contents of the line clicked on.

Table 2

This describes the handlers in the "Fixed Assets" stack script.

clicking on it. The depreciation is plotted year by year, and the method is displayed in the "Depreciation Method" field. The values at the left of the graph are calculated based on the highest value in the table to be plotted. The scale at the bottom of the graph always reads 0 to 50 years.

Clicking on the "Clear" button in the lower-left corner of the card clears the graph area. The "Grid" button overlays a grid on the graph to make it easier to read. Click on the "Grid" button again to make the grid disappear. The "Print" button prints the current screen to the printer. To return to the main screen for your asset, click on the return arrow in the lower-left corner.

The Stack Script

The key to this stack lies in the stack script that performs the depreciation calculations. These scripts can be used to create your own applications, or you may wish to obtain the *StackSolutions* disk from *HyperLink Magazine* which contains the Fixed Asset stack in its entirety.

The stack script, reproduced in its entirety in Listing 1, contains the bulk of the scripts for this stack. Table 2 gives a brief explanation of what each of the various handlers do in the stack. If you do try to use these scripts make careful note of how the global variables are being used.

—Listings continue on next page

Listing 1 - cont.

```
 global Fm,Lm,Fyear,Lyear,RealAge,flag2
 global Per1 AqCost, Origin,Graph
 put empty into Graph
 put 0 into Total
 ask "Decining Rate (100 - 200%)" with ¬
 field "Dec Rate"
 if it < 100 or it > 200 then
   ask "Enter a rate from 100 to 200:" ¬
   with field "Dec Rate"
   if it < 100 or it > 200
   then exit Declining_Balance
 end if
 put it into fld "Dec Rate"
 put it into DecRate
 SetUp
 if not Flag2 then exit Declining_Balance
 put Book into AdjBook
 put (1 / Life) * (DecRate / 100) into ¬
 Rate
 put CheckNum(AdjBook * Rate * Fm) into ¬
 line 1 of Graph
 put (AdjBook * Rate) - line 1 of Graph ¬
 into HoldOver
 add line 1 of Graph to Total
 subtract Total from AdjBook
 repeat with x=2 to trunc(Life)
   set cursor to busy
   put HoldOver + (AdjBook * Rate * Fm) ¬
   into Dep
   put (1-Fm) * AdjBook * Rate into ¬
   HoldOver
   if x = trunc(Life) then
     put CheckNum(Book - Total) into ¬
     line x of Graph
     exit repeat
   end if
   add Dep to Total
   subtract Dep from AdjBook
   put CheckNum(Dep) into line x of Graph
 end repeat
 if Origin is "Opt1" then put Graph ¬
 into fld "Opt1 Values"
 if Origin is "Opt2" then put Graph ¬
 into fld "Opt2 Values"
 if Origin is "Opt3" then put Graph ¬
 into fld "Opt3 Values"
end Declining_Balance

on Declining_Straight
 global AssetAge,Life,Book,Lmonth,Fmonth
 global Fm,Lm,Fyear,Lyear,RealAge,flag2
 global Per1 AqCost,Origin,Graph
 put empty into Graph
 put 0 into Total
 ask "Decining Rate (100 - 200%)" with ¬
 field "Dec Rate"
 if it < 100 or it > 200 then
   ask "Enter a rate from 100 to 200:" ¬
   with field "Dec Rate"
   if it < 100 or it > 200 then exit ¬
   Declining_Balance
    end if
   put it into fld "Dec Rate"
   put it into DecRate
   SetUp
   if not Flag2
   then exit Declining_Straight
   put Book into AdjBook
   put (1 / Life) * (DecRate / 100) ¬
    into Rate
   put CheckNum(AdjBook * Rate * Fm) ¬
   into line 1 of Graph
   put (AdjBook * Rate) - line 1 of ¬
   Graph into HoldOver
   add line 1 of Graph to Total
   subtract Total from AdjBook
   repeat with x=2 to trunc(Life)
   set cursor to busy
   put HoldOver + (AdjBook * Rate * Fm) ¬
   into Dep
   put (1-Fm) * AdjBook * Rate into ¬
   HoldOver
   if AdjBook / (Trunc(Life) - x + 1) > ¬
   Dep then exit repeat
   add Dep to Total
   subtract Dep from AdjBook
   put CheckNum(Dep) into line x of Graph
   end repeat
   put trunc(Life) - x + 1 into RemLife
   repeat with x=x to trunc(Life)
   set cursor to busy
   put CheckNum(AdjBook / RemLife) ¬
   into line x of Graph
   add word 1 of line x of Graph to Total
   end repeat
   if Book > Total then put Book - ¬
   Total into line x + 1 of Graph
   if Origin is "Opt1" then put Graph ¬
   into fld "Opt1 Values"
   if Origin is "Opt2" then put Graph ¬
   into fld "Opt2 Values"
   if Origin is "Opt3" then put Graph ¬
   into fld "Opt3 Values"
  end Declining_Straight

on Sum_Of_Years_Digits
 global AssetAge,Life,Book,Lmonth,Fmonth
 global Fm,Lm,Fyear,Lyear,RealAge,flag2
 global Per1,AqCost,Origin,Graph
 put empty into Graph
 SetUp
 if not Flag2
 then exit Sum_Of_Years_Digits
 put 0 into SumTotal
 repeat with x=1 to trunc(Life)
   set cursor to busy
   add x to SumTotal
 end repeat
 put 0 into Total
 put CheckNum((Trunc(life) / SumTotal) ¬
 * Book * Fm) into line 1 of Graph
 put line 1 of Graph into Total
 repeat with x=2 to trunc(Life)
   set cursor to busy
   put ((Trunc(Life) - x + 1) / ¬
   SumTotal) * Book into Dep
   add Dep to Total
   put CheckNum(Dep) into line x of Graph
 end repeat
 if Total < Book
  then put CheckNum(Book - Total) ¬
 into line x + 1 of Graph
 if Origin is "Opt1" then put Graph ¬
 into fld "Opt1 Values"
 if Origin is "Opt2" then put Graph¬
 into fld "Opt2 Values"
 if Origin is "Opt3" then put Graph ¬
```

Listing 1 - cont.

```
 into fld "Opt3 Values"
end Sum_Of_Years_Digits

on Units_of_Output
  global AssetAge,Life,Book,Lmonth
  global Fmonth,Fm,Lm,Fyear,Lyear
  global RealAge,flag2 Per1,AqCost,Origin,
  global Graph
  if Origin is "Graph" then go back
  put empty into Graph
  if Origin is "Opt1" then put fld ¬
  "Opt1 Values" into Graph
  if Origin is "Opt2" then put fld ¬
  "Opt2 Values" into Graph
  if Origin is "Opt3" then put fld ¬
  "Opt3 Values" into Graph
  SetUp
  if not Flag2 then exit Units_Of_Output
  put 0 into Total
  Ask "Total units into the life of" && ¬
  "this asset:" with field "Units"
  if it is empty then exit Units_Of_Output
  put it into field "Units"
  repeat with x=1 to the number of ¬
  lines of Graph
    set cursor to busy
    put checkNum(item 1 of line x of ¬
    Graph / it * Book) into item 2 of ¬
    line x of Graph
  end repeat
  if Origin is "Opt1" then put Graph ¬
  into fld "Opt1 Values"
  if Origin is "Opt2" then put Graph ¬
  into fld "Opt2 Values"
  if Origin is "Opt3" then put Graph ¬
  into fld "Opt3 Values"
end Units_of_Output

on Calculate
  global AssetAge,Life,Book,Lmonth,Fmonth
  globalFm,Lm,DepValues,Fyear,Lyear
  global RealAge,flag2,Per1,AqCost
  global Origin,Graph
  Setup
  if Flag2 is false then exit Calculate
  if RealAge <= 0 then
    put 0.00 into fld "Depreciation"
    put 0.00 into fld "Total Depreciation"
    exit Calculate
  end if
  if AssetAge > 50 then put 50 into ¬
  AssetAge
  if RealAge < Fm then
    put CheckNum(line 1 of DepValues * ¬
   RealAge into fld "Depreciation"
    put fld "Depreciation" into fld ¬
    "Total Depreciation"
    exit Calculate
  end if
  put 0 into Total
  repeat with x=0 to AssetAge - 1
     set cursor to busy
    add line x + 1 of DepValues to Total
  end repeat
  get field "Dep Method"
  if it is not "Units_of_Output" then
    if AssetAge = trunc(Life) then
      put (RealAge - trunc(RealAge)) + ¬
     Fmonth into LRmonth
      if LRmonth > 12 then subtract 12 ¬
      from LRmonth
      if Lmonth >= LRmonth then
        put CheckNum(line x + 2 of ¬
       DepValues) into fld "Depreciation"
      else put CheckNum(Lmonth / LRmonth ¬
      * line x + 2 of DepValues)¬
      into fld "Depreciation"
    else put CheckNum(Lm * line x + 2 of ¬
    DepValues) into fld "Depreciation"
  else put CheckNum(line x + 2 of ¬
  DepValues) into fld "Depreciation"
  put fld "Depreciation" + Total into ¬
  fld "Total Depreciation"
end Calculate

on Manual
  -- this routines intercepts
  -- the "Manual" message
end Manual

on Finish
  global AqCost
  put CheckNum(AqCost - fld ¬
  "Total Depletion" - fld ¬
  "Total Amortization" - fld ¬
  "Total Depreciation") ¬
  into fld "Book Value"
  if fld "Actual Value" is not empty
  then put fld "Actual Value" - fld ¬
  "Book Value" into fld "Gain/Loss"
end Finish

on FinishAmortization
  global AssetAge
  Age
  if fld "Amortization List" is empty then
    put 0.00 into fld "Amortization"
    put 0.00 into fld "Total Amortization"
    exit FinishAmortization
  end if
  put CheckNum(line (AssetAge + 1) of ¬
  fld "Amortization List") into ¬
  fld "Amortization"
  put 0 into Total
  repeat with x=1 to AssetAge + 1
    add line x of fld ¬
    "Amortization List" to Total
  end repeat
  put CheckNum(Total) into fld "Total
Amortization"
end FinishAmortization

on FinishDepletion
  global AssetAge
  Age
  if fld "Depletion List" is empty then
    put 0.00 into fld "Depletion"
    put 0.00 into fld "Total Depletion"
    exit FinishDepletion
  end if
   put CheckNum(line (AssetAge + 1) of ¬
  fld "Depletion List") into fld ¬
  "Depletion"
  put 0 into Total
  repeat with x=1 to AssetAge + 1
    add line x of fld "Depletion List" ¬
    to Total
```

Listing 1 - cont.

```
 end repeat
  put CheckNum(Total) into fld ¬
  "Total Depletion"
end FinishDepletion

on FinishUnits
  global AssetAge,DepType
  Age
  if DepType is empty
  then exit FinishUnits
  if line AssetAge + 1 of fld DepType is¬
   empty then
    put 0 into fld "Units Used"
  else put item 1 of line AssetAge + 1 ¬
  of fld DepType into fld "Units Used"
end FinishUnits

on SetUp
 global AssetAge,Life,Book,Lmonth,Fmonth
 global Fyear,Lyear,RealAge,flag2,AqCost
  global Source
  Age
  if flag2 is false then exit SetUp
  put (((Lyear*12)+Lmonth) - ¬
  ((Fyear*12)+Fmonth)) / 12 into RealAge
  put fld "Years to Dep" into Life
  put fld "Aquisition Cost" into AqCost
  if AqCost is empty then
    ask "Enter acquisition cost of asset:"
    if it is empty then
      put false into Flag2
      exit Setup
    end if
    put it into AqCost
    put it into fld "Aquisition Cost"
  end if
  put AqCost into Book
  if hilite of bg btn "Use" and Source ¬
  is not "Declining_Balance"¬
  and Source is not "Declining_Straight"¬
  then subtract fld "Salvage Value" ¬
  from Book
end SetUp

on Age
  global Lyear,Lmonth,Fyear,Fmonth,
  global AssetAge,Fm,Lm,flag2
  put true into flag2
  put fld "Date Aquired" into FDate
  if FDate is empty then
    ask "Aquisition date of asset:"
    if it is empty then
      put "false" into flag2
      exit Age
    end if
    put it into FDate
    put it into fld "Date Aquired"
  end if
  put fld "Date of Value" into LDate
  if LDate is empty then
    ask "Valuate this asset to what
date:"¬
   with the date
    if it is empty then
      put "false" into flag2
      exit Age
    end if
    put it into LDate
    put it into fld "Date of Value"
  end if
  convert LDate to DateItems
  convert FDate to DateItems
  put item 1 of LDate into Lyear
  put item 2 of LDate into Lmonth
  put Lmonth / 12 into Lm
  put item 1 of FDate into Fyear
  put item 2 of FDate into Fmonth
  put (13 - Fmonth) / 12 into Fm
  put max(0,Lyear - Fyear) into AssetAge
end Age

on Graphit
  global Graph
  if Graph is empty then
    exit Graphit
  end if
  set numberformat to "0"
  put 0 into High
  repeat with x=1 to the number of lines ¬
  of Graph
    set cursor to busy
    if line x of Graph > High
    then put line x of Graph into High
  end repeat
  put the number of chars of ¬
  trunc(High) into Len
  put (char 1 of High + 1) * ¬
  10^(Len - 1) into High
  repeat with x=1 to 10
    set cursor to busy
    put (High * x) / 10 into line ¬
    11 - x of fld "Value"
  end repeat
  repeat with x=1 to the number of ¬
  lines of Graph
    set cursor to busy
    put trunc(69+(x*8)) into item 1 ¬
    of line x of Graph2
    put trunc(243 - (line x of Graph /¬
    High * 216)) ¬
    into item 2 of line x of Graph2
  end repeat
  ClearScreen
  doMenu "Background"
  choose Line tool
  repeat with x=2 to the number of ¬
  lines of Graph2
    set cursor to busy
    drag from line x-1 of Graph2 to line¬
    x of Graph2
  end repeat
  choose browse tool
end Graphit

on ClearScreen
  set cursor to busy
  lock screen
  domenu "Background"
  choose select tool
  drag from 69,27 to 505,243
   doMenu "Cut Picture"
  choose browse tool
end ClearScreen

function CheckNum Num
  set the numberformat to "0.00"
  put 0 into decimals
```

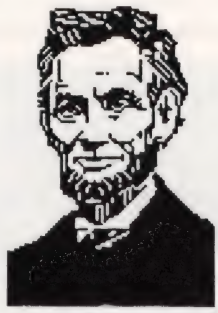
Listing 1 - cont.

```
 put 0 into minuses
  repeat with curnum = 1 to length(Num)
    set cursor to busy
    if char curnum of num = "." then add 1 to decimals
    if char curnum of num = "-" then add 1 to minuses
    if char curnum of num is in "-1234567890." and¬
    decimals < 2 and minuses < 2 then next repeat
    else
      return "Error"
      exit CheckNum
    end if
  end repeat
  if num = "." or num = "-" or num = ".-" or num ="-." then
    return "Error"
    exit CheckNum
  end if
  if Num < 0 then put 0 into Num
  add zero to Num
  put round(Num * 100) / 100 into Num
  return Num
end CheckNum

function ClickLine Type
  if style of target is "Scrolling" then
    put trunc((item 2 of the clickloc - the top of target ¬
    + (scroll of target) / textheight of target) +1 into Ln
  else put trunc((item 2 of the clickloc - ¬
  the top of target) / textheight of target) + 1 into Ln
  if Type is empty then return Ln
  else return line Ln of Target
end ClickLine
```


"When you have got an elephant by the hind leg, and he is trying to run away, it's best to let him run."

Abraham Lincoln, 1865



With ***Presidents***, students will run straight into a wealth of information on 40 American Presidents.

Using HyperCard,™ facts on each president's election, administration, political career, personal history, family, birth, death, quotes, and more are discovered through ten stacks of information.

Presidents is an excellent resource for teaching history and political science.

By Alice Jaggar, Tom Layton, and Bob Veeck of Data Disk International.

Single User: $ 49.95
Site License: $ 149.95

To order, contact:
ICCE
1787 Agate Street
University of Oregon
Eugene, OR 97403
(503) 686-4414
HyperCard not included

—continued from page 12

```
SELECT LAST_NAME, FIRST_NAME,
COMPANY
FROM CUST
WHERE CUST_TYPE = 'REG'
OR COMPANY = 'Apricot Press'
ORDER BY LAST_NAME;
```

Multiple-Table Retrieval

As you can see from what we've already discussed, SQL offers powerful data-retrieval capability. And we haven't really begun to explore that subject in depth; for that, you need a book on SQL. There are several good ones available, but the best one I've found is *Mastering ORACLE* by Daniel J. Cronin, a senior Oracle technical staff member.

Even if SQL could only retrieve information flexibly from a single table, it would be a wonderful extension to *HyperCard's* relatively limited data-management capabilities. When it comes to combining information from two or more related tables, *ORACLE* and SQL really shine.

In what we have done so far, the delivery person in our deli would not necessarily know where to go to deliver an order because the CUST table doesn't contain addresses. Instead, it has only company names and, in some cases, building numbers. Remember that we designed the table that way intentionally, so we could avoid storing duplicate addresses in the table. This data-space conservation is one of the major advantages of a relational database model like *ORACLE*.

Let's now assume that we've filled out the ADDRESSES table and that its contents are as shown in Figure 3. Jerry Allen calls with an order for lunch, and we want to print the company name and address for the delivery person. Here's how the SQL query would look (I'll explain it shortly):

```
SELECT LAST_NAME, FIRST_NAME,
COMPANY, BUILDING,
ADDRESSES.ADDRESS
FROM CUST, ADDRESSES
WHERE CUST.COMPANY = ADDRESS-
ES.COMPANY
AND CUST.BUILDING = ADDRESS-
ES.BUILDING;
```

(Notice that the ORDER BY clause has been eliminated because we expect this query to return only one result and therefore need not sort the output.)

Company	Building	Address
National Can	42	2345 Santa Cruz Ave.
National Can	43	15549 Tenth St., Suite 44
National Can	44	10 Main St.
Delphi Corp.		111 Fort Rd., Suite J
Smythe Incline		277 Hillbarn Avenue
Apricot Press		994 Elmer Fudd Way

Figure 3
ADDRESSES Table Contents

Customer	Type	Definition
OCC	Occasional	Buys less than once a month
REG	Regular	Buys at east once a month
CONF	Conference	Not regular but large orders when placed
DORM	Dormant	No order in at least six months

Figure 4
CUST_TYPE Table Contents

Because we are working with two tables, we must tell SQL in the SELECT statement when we are instructing it to extract data from a table other than the "main" one. We do this by appending the name of the secondary table(s) to the front of the name(s) of the column(s) to extract and separating them by a period. Thus the construction ADDRESSES.ADDRESS is interpreted by SQL to mean "the ADDRESS column from the table ADDRESSES." In the FROM clause, the first table we list is assumed by SQL to be the "main" table to which it will default if we don't give it a specific table name from which to select a particular column.

The key to a multiple-table retrieval—technically referred to as a JOIN operation in relational database terms—is the presence in both tables of a key field or fields that have common values. In most cases, this will be a single key field, but as you can see from our example, that need not be the case. In the example, we've used the equality of two separate columns as the basis for combining the information from the two tables.

Here's another example of a multi-table JOIN operation. This time, we're using the contents of the CUST_TYPE table shown in Figure 4 to produce a list that shows each customer's name and type, where the type is spelled out rather than being displayed as a cryptic code. This is a common use of the multi-table retrieval technique.

```
SELECT LAST_NAME, FIRST_NAME,
CUST_TYPES.DEFINITION
FROM CUST, CUST_TYPES
WHERE CUST.CUST_TYPE =
CUST_TYPES.CUST_TYPE
ORDER BY LAST_NAME;
```

Notice here that the columns on which we are asking SQL to match the two tables—the CUST_TYPE column in each table—are not even among those we are retrieving. This is perfectly permissible in a SELECT statement. Also note that although in our examples the corresponding columns in the two tables had the same names, this is neither necessary nor always desirable. The query above, for example, might be easier to read if we had chosen more uniquely descriptive names for our tables and columns.

There's Much More

We have only scratched the surface of the capability of SQL in this series of quick looks at the language. Vast stretches of the language remain for you to explore on your own. But I hope that what we have been able to explore together has convinced you that SQL is a powerful language that can be used in conjunction with *HyperCard's* great graphic interface, programming ease, and ease of use to provide front ends to SQL-type databases. Insulating the deli owner from the necessity of learning and then typing all of these SQLcommand strings—and we have not really explored some of the more complex ones she'd have to use to run the business well—is a great boon of *HyperCard* in this setting.

Speculation About a HyperCard *Workalike from IBM Is Finally Answered... But How Does It Stack Up*

A Look at IBM LinkWay

by Larry Burtness

IBM users have watched the growth in popularity of *HyperCard* as a development system for information management applications and speculated on the adaptation of *HyperCard* for the IBM PC. IBM recently announced a product called *IBM LinkWay* that is billed as a hypertext tool for organization of information. It is capable of providing ways of linking information that is presented in text, pictures, video, and other forms. Although it is not a *HyperCard* clone, *LinkWay* brings many of the concepts and capabilities of *HyperCard* to IBM users.

LinkWay will inevitably be compared with *HyperCard*, and although there are some similarities, there are also many differences. Here, we will explore some of these similarities and differences and provide you with an in-depth look at this new tool.

LinkWay System Configurations

LinkWay was designed with an eye toward memory conservation throughout its development. The software requires, as the manual puts it, "the appropriate version of DOS for your computer," a minimum system configuration of 384K of memory, a color graphics display, an IBM or Microsoft compatible mouse controller, and at least one floppy drive. Because the majority of IBM PC's and compatibles in use have 640K of RAM, most machines have the required memory to run the program.

An optimum configuration for a PC running *LinkWay* would include 640K of memory or more, a hard disk, and an MCGA or VGA video graphics adapter (see Figure 1). These video displays provide good resolution, with up to 256 colors available on the screen. Other graphics configurations will also work, with *LinkWay* recognizing and using the highest resolution display that is available on the system when it is started up. If it is available, memory above 640K can be used as a RAM Disk to speed up operation of the system.

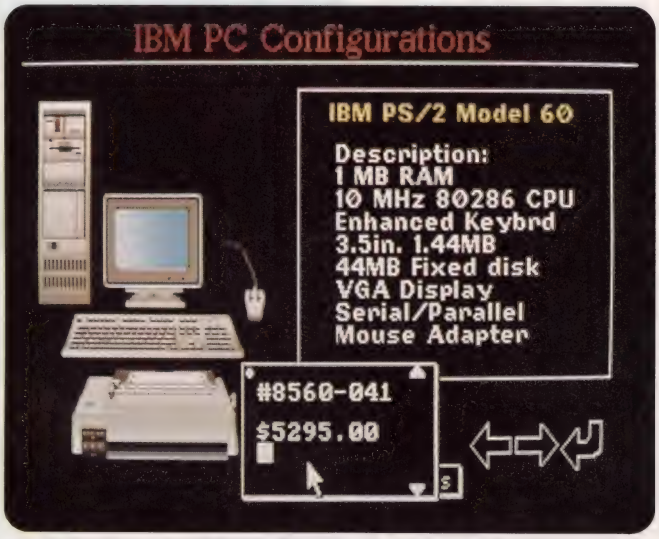


Figure 1

Although LinkWay *will run on any PC or compatible with a minimum of 384K of RAM, this* LinkWay *page shows a system ideally suited to running the new* IBM LinkWay *software.*

IBM LinkWay Software Tools

The *IBM LinkWay* system includes the *LinkWay* program and a number of utility files for accomplishing a variety of tasks. The *LinkWay* program is about 80K in size and provides the user with the overall operating environment. This includes the ability to create and to execute *LinkWay* application "folders," the name given to *LinkWay* files. A *LinkWay* folder is roughly the equivalent of a *HyperCard* stack. The program works via an interface of pull-down menus, on-screen buttons, and pop-up menus, which are all mouse selected.

Among the other utility programs and sample applications included with *LinkWay*, are a paint program, a text editor, a font editor, and some demonstration folders such as a calendar and a "to do" list.

The paint program, called *LWPaint*, is a simple painting program with basic drawing tools for creating and editing pictures. *LWPaint* works in any of the graphics screen formats supported by *LinkWay*, automatically using the screen format that matches your

Larry Burtness is the Director of the Educational Technology Center for Educational Service District #189 in Mount Vernon, Washington. He specializes in training teachers to apply computers in education. He has been teaching HyperCard *since its introduction, and using pre-release versions of* IBM LinkWay *for over a year.*

computer's hardware. The program can work with files created with *PC Paintbrush* (.PCX files) and *LinkWay* includes a screen capture program called *LWCapture* that allows the capture of any screen into a file which is compatible with *LWPaint* and *LinkWay*. Incorporating graphic images into *LinkWay* applications can be accomplished using a variety of paint programs, video image capturing boards, or scanners for the PC (see Figure 2).

LWEdit is a text processor that is included with the *LinkWay* system, and it can be incorporated as an integrated part of any *LinkWay* application. The editor may be used to display text files that contain resources for the application folder, or it may be used as a rudimentary processor for work by the application user. *LWEdit* is a basic word processor, lacking the features of more professional word processors. It does allow easy text entry and editing, margin formatting, simple block text moves and printing of documents from the application. In its current form, the editor is not mouse based, using function keys to invoke the various features of the program, with a built-in help screen available at the press of the IBM's F1 key.

LWFontEd is a font editing program that allows any of the font or icon tables for the *LinkWay* system to be edited and modified. The most common use of this will be to add custom icons to the icon file which applications use. The editor is simple to use, allowing "fat-bit" style manipulation of pixels on the icon or font characters.

Each of these utilities allows the application author to enhance *LinkWay* applications for specific uses without relying on other applications. Separate, more powerful applications can be used effectively in place of the provided applications if you wish. For example the popular paint programs *Splash!*, from Spinnaker, and *Deluxe Paint II*, from Electronic Arts, work well for creating and editing *LinkWay* graphics and offer far more features than *LWPaint*.

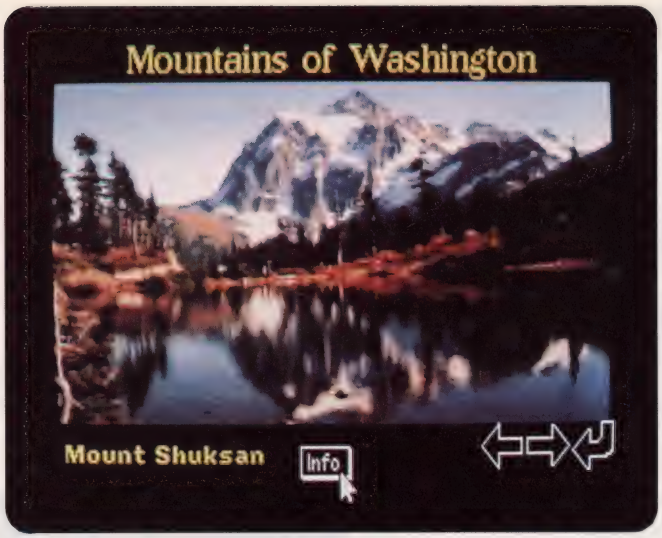


Figure 2

Paint graphics are stored externally and are read into LinkWay *folders into user defined picture blocks.This 256 color image was captured using a scanner.*

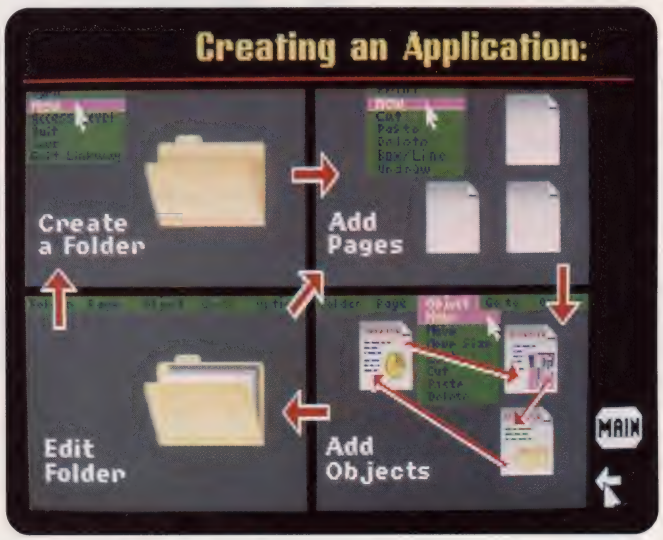


Figure 3

The basic organization of a LinkWay *application and how it is developed is shown on this* LinkWay *page.*

LinkWay Data Organization

Data organization in *LinkWay* is similar in many ways to *HyperCard*, however there are some major differences. Many of the differences are due to the nature of the video display on the IBM PC, with the overall data structure oriented around conserving memory space as much as possible. Features were left out in effort to keep the system as small as possible. This memory saving allows the program to run using small capacity computers, and to run quite quickly for most uses. *LinkWay* cannot, however, hope to match all of the features of a program such as *HyperCard*, which weighs in at 380K plus.

The information in the folder (See Figure 3) is organized on pages, similar to what *HyperCard* refers to as cards. The information that is placed on a page is in data fields, in buttons of various types, or in pictures. Each of these objects has various attributes that dictate its appearance and function in the folder.

The *LinkWay* folder is read from disk into memory when it is accessed. The entire folder, less pictures and text documents, must fit in memory. *LinkWay* makes provision for reserving memory space from 4K to 250K for folders, with memory that is left over available for running external applications or utilities. If the information to be stored in *LinkWay* folders is greater than 250K, it must be broken into separate folders, with links established between the folders. Any number of folders may be linked in this way, so the actual data capacity is limited only by the size of the disk drive.

HyperCard uses backgrounds to contain elements that are to common to all cards in a stack, *LinkWay* uses a "Base Page" to contains objects that act in a similar way. Buttons on the Base Page are visible and accessible from all pages in the folder. Background fields in *HyperCard* act as unique fields on all cards in a stack, in *LinkWay*, however, fields that are on the base page contain the same information on all pages. Thus, the most common use of fields on the Base Page in *LinkWay* is as an identifying label, and they are not appropriate for data

that is changed from page to page.

Unlike *HyperCard* where a picture layer is part of the card, pictures in *LinkWay* are not stored as a part of the page on which they are displayed. They are stored in separate picture files, and read from the disk when they are required for display. Pictures are placed on the page in a layer below the fields and buttons on the page, so the fields and buttons appear on top of a picture.

Pages are assigned ID numbers as they are created in any particular folder, beginning with 1 and counting upward. ID numbers are always unique within a folder, and are not repeated, even if a page is deleted. In addition to an ID number, each page is assigned a sequence number, which is the ordered position of the page in the folder. The sequence number may change as new pages are added to, subtracted from, or moved around within the folder, but the ID number remains permanent. Both ID and sequence numbers can be used as destinations of links within the *LinkWay* system. *LinkWay* assigns unique ID numbers only within a folder, not across folders. The ID number of the Base Page of any folder is always zero.

Objects that are placed on a page in *LinkWay* are placed in character-mapped positions (every 8 pixels), rather than bit-mapped positions as in *HyperCard*. This allows Fields, Buttons, and Pictures to be placed only on character location boundaries, not pixel by pixel. This simplification in the system allows for somewhat less flexibility for some requirements, but makes the on-screen alignment of objects easy.

Five Different Access Levels

User access to *LinkWay* folders is controlled by a password protected access level assignment. The assignment levels are Read, Update, Insert, Delete, and Format. Creating and editing objects in a folder is possible only when the access level of a folder is set to Format. A Read level access allows read-only access, data is not modifiable by the user. Update access allows fields that are not locked to be changed. Insert and Delete access levels refer to the ability of the user to add or remove pages from the folder. Developers who are distributing application folders for *LinkWay* can provide the level of access that is appropriate for the application.

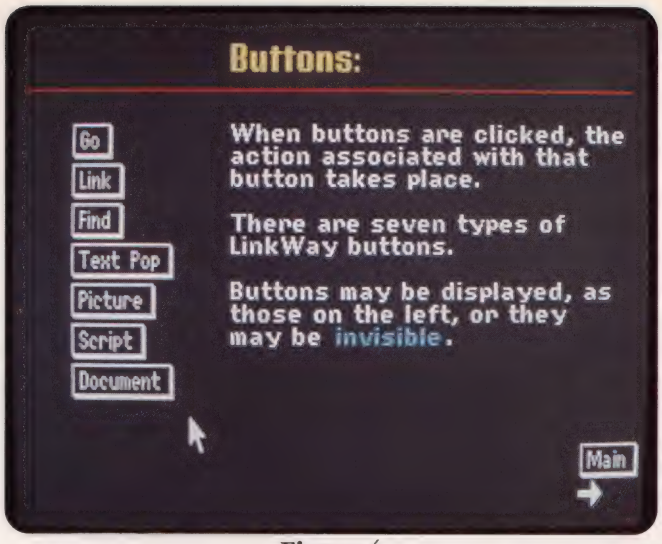


Figure 4
This screen displays the seven button types available within a LinkWay *application.*

Buttons

Buttons in *LinkWay* are used in a fashion similar to those in *HyperCard*—as the operative objects for controlling the functions of the application. There are seven different types of buttons, each having a different use (see Figure 4). By carefully employing the different types, many applications can be developed without any scripting.

The button types are Go, Link, Find, Text pop-up, Picture pop-up, Script, and Document. The only one that allows the author to write script code is the Script type, all others have built-in functions that the author assigns from a series of menus and dialog boxes as the button is created.

HyperCard uses a separate tool

for each type of object, but *LinkWay* does not differentiate between the creation of a Button, Field, or Picture object. All are created from the same pull down menu, using the same techniques.

When you create a new button, you select the Object menu, pulling down to the "New" option. An on-screen pop-up menu allows you to select the object type to be created, either a Button, Field, or Picture. After the type is selected you drag the object to the desired position and size.

After positioning the button, the Button type is selected from a menu of the seven different types. After selecting the button type, a dialog box asking for the button name appears, this is optional and may be passed by, or you may enter a name up to eight characters in length.

A dialog box for selecting the desired icon to be used is then displayed. You have a choice of 96 different icons, and they may be modified as desired using the *LWFontEd* program.

After selecting the icon for the "Go" button, you select the destination—either Base, First, Last, Next, or Previous—from an on-screen menu. You create each object in a similar fashion selecting its parameters and attributes.

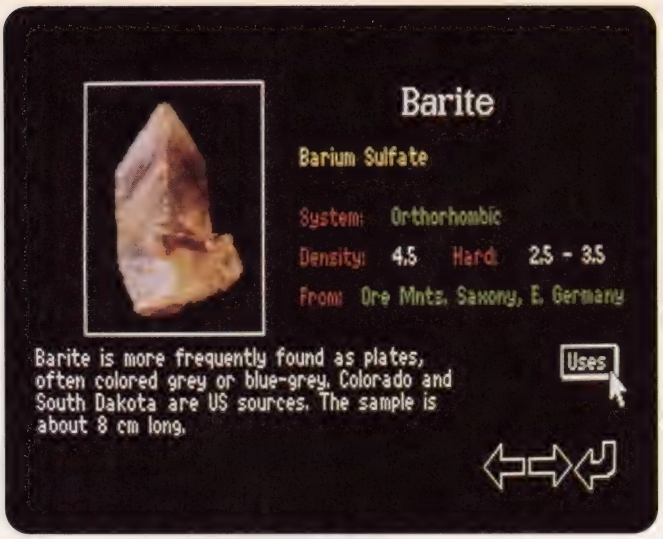


Figure 5

This screen shows how the combination of colored text and graphics make LinkWay *ideally suited to education.*

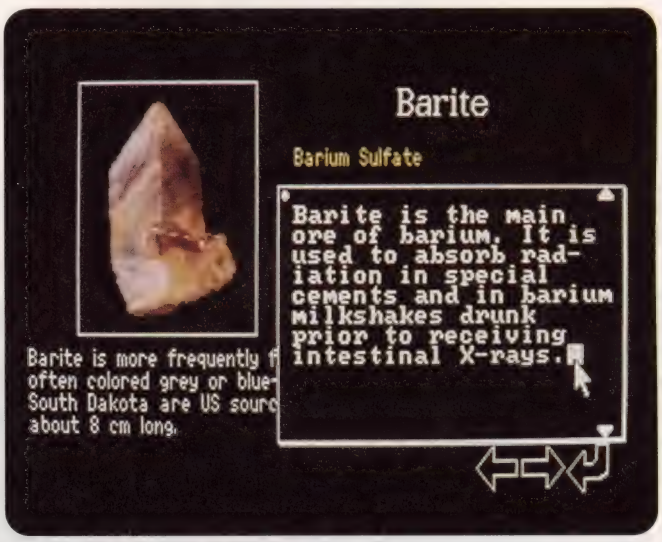


Figure 6

Pressing on the "Uses" button above, brings up a Text pop-up field with added information.

Special Button Types

There are two special button types that do not exist in *HyperCard*, the text pop-up and the picture pop-up. The text pop-up allows you to place a button on a page that pops-up to become a small scrolling text window (see Figures 5 and 6). This can contain up to 3000 characters of text with scrolling controls similar to scrolling fields in *HyperCard*. A similar function can be set up in *HyperCard* using a button to control a scrolling field on a card, using HyperTalk's show field command. *LinkWay*'s pop-up text buttons are easier to implement than the equivalent in *HyperCard*, however, the *HyperCard* scrolling field is capable of holding much more data, 32K versus 3K.

Picture pop-ups are similar to text pop-ups except that a part of a graphics picture is placed on the page, rather than a field of text. The picture may be a piece of a larger graphic produced by a paint program or a scanning system. The picture may be re-sized and repositioned on the screen as desired. Using extension programs to *LinkWay*, which are a part of the *LinkWay Toolkit*, the pop-up Picture may be an animation sequence as well. Pop-up pictures are not a part of *HyperCard*, but have been implemented as XCMD's. Implementation of this type of action in *HyperCard* without the use of an external command requires a change to a second card with the pop-up picture on itt

A document button is also a unique button type that has the express purpose of invoking the document editor, *LWEdit*. This editor is capable of loading and editing files that are up to about 25K in size. Document buttons can be brought up with text files already in them, or

Scripts in Script Buttons Only

Script buttons allow the author of a folder to create buttons for special purposes, programming in a simple scripting language. The language is different from HyperTalk, the language of *HyperCard*, and does not contain as many features and functions. See the "Table of *LinkWay* Script Statements" at the end of this article for a list of all built-in scripting commands.

The language allows program structuring through the use of IF...ELSE statements, and jumps to labeled locations in a script. There are no Repeat statements, but these may be created using a combination of IF...ELSE statements and counter variables. Page to page or folder to folder links, finds, cutting and pasting of pages, printing and other script statements allow the script programmer to control the access to pages in folders under script control.

The script programmer can provide interaction with the user through input statements, prompting messages, and pop-up on screen menus through script statements. In addition, data can be read and written to and from disk files.

The scripting language allows for the execution of any DOS command from within a script, as well as the extension of the system through the use of external routines.

with the editor empty, ready for the users text entry. These can be used for providing supplementary text information to the on-screen fields and other data in the folder

The document text files are stored outside of the *LinkWay* folder, just as pictures are. They do not add to the size of the folder, but do require disk storage space. They can be very helpful in allowing documents to be accessed quickly from within *LinkWay*, for example, to place a text file containing the documentation of an application directly in the application by the inclusion of a document button for the documentation file.

Fields for Data

Fields in *HyperCard* can have many different forms and visual attributes. Fields may also have HyperTalk scripts associated with them that can cause the field to perform some programmed operation. In *LinkWay*, fields can be set up to hold text or numeric data, but the on-screen visual attributes that can be controlled are fewer. They include the size and screen placement of the field, the color of the text, and the size and style of the font used in the field.

LinkWay fields cannot have scripts that are directly held by the field. This can be implemented in *LinkWay* by placing an invisible Script Button over the top of the field that refers to the field in its script. Script buttons are the only objects in *LinkWay* that can contain script code. Any object in *HyperCard* can have a HyperTalk script associated with it.

Running Other Applications From *LinkWay*

LinkWay has a number of provisions for extending the capability of the system and running other application programs from within a *LinkWay* folder. There are three separate ways to run external programs, each with different purposes and each increasing the power of *LinkWay* for custom applications. Some of these are similar to the way in which *HyperCard* is extended using XCMD's.

Assembler or binary routines for specific tasks can be run from a *LinkWay* script button. Programmers familiar with techniques of assembly programming can use this capability to create specific drivers or special adaptations.

In addition, *LinkWay* has provision for running a custom subroutine written in C or some other system programming language. The external routine can pass data to and from *LinkWay* folders, even access the data pages in a *LinkWay* folder directly. This kind of access should only be attempted by experienced programmers, because data in the folder can be destroyed if it is done improperly. This capability does allow *LinkWay* applications to be developed well beyond the native capabilities of the system, and should be of great interest to developers of applications.

In addition to being able to run user-written assembler and other routines, *LinkWay* has the capability of running any DOS command by using a DOS "shell." Any application that is run under DOS should be able to be run from *LinkWay* as long as it fits in the available memory. This can be used for a number of different purpose.

Table of LinkWay Script Statements

Variable Handling Statements

```
VAR      declares and allocates memory for variables
SET      assign values to variables
LOAD     read a file directly into a variable
VCLEAR   discard all active variables
```

Control Statements

```
REMARK   /* anything between a "/* */" is a remark
         */ and ignored
LABEL    @NAME is used to mark a program point
JUMP     JUMP NAME jumps to a labeled point
IF/ELSE  allows conditional statements
EXTERN   call an external program
SCRIPT   call and execute another script
BCALL    call a binary routine
DOS      execute a DOS command
MODE     hange video display mode
QUIT     terminate and return to DOS
```

Page Handling Statements

```
GO       go to a specified page number
LINK     link to a folder and page
FIND     search for a data match on a page
RETRACE  return to the top page of the retrace stack
CUT      copy a page to a disk file
PASTE    paste a page into folder from a disk file
CLONE    duplicate the current page
DELETE   delete the current page
CPRINT   print a graphic image of the current page
SHOW     update the screen on page change
NOSHOW   do not update the screen on page change
MBAR     display the menu bar
NOMBAR   hide the menu bar
```

Object Handling Statements

```
DO       activate a field as if user selected
OBJECT   select an object for later reference
MOUSE    wait for mouse action
```

Input/Output Statements

```
MSG      display a message box on the screen
PROMPT   display a prompt line at bottom of screen
MENU     display a pop-up menu on the screen
INPUT    prompt and receive input from keyboard
PRINT    print a line on the system printer
SERIAL   send data to serial port
READ     read a data file from the disk
WRITE    write a data file to disk
WAIT     wait a specified number of seconds
BEEP     beep once
```

Video Display Modes

One of the difficulties of creating application programs for the IBM PC comes from one of the strengths of PC's and compatible systems—the diversity of video display modes available. There are color graphics display modes available in many resolutions and capacities, for any need and any budget. *LinkWay* supports all of the color display modes used by IBM on its personal computers, including 4 color CGA (320 X 200 pixel), 16 color EGA and VGA (640 X 400 pixel), 256 color MCGA (320 X 200 pixel), and High Resolution Monochrome (640 X 400 pixel). The program recognizes the highest display mode your system is equipped with and attempts to use that display.

Converting applications from one mode to another is quite easy, file extensions are changed to reflect the new mode desired, and the folder is opened using the new display mode. *LinkWay* automatically converts the folder to the new display mode. Some object placement will not be exact, especially in changing from 320 X 200 to 640 X 350 display modes, so some editing of object positions may be necessary, but the folder remains fully functional.

Special Features

LinkWay provides capability to access a number of special peripheral devices, as well as the normal resources of the computer. The scripting language includes the SERIAL statement allowing access to the serial port for communications or control of devices such as laser videodisc players. Folders with videodisc related data and interactive control are relatively easy to create and provide exciting possibilities.

Special applications using IBM *InfoWindows* displays, IBM speech options, music options, and other peripheral hardware are also possible. The main IBM marketing efforts for *LinkWay* thus far are in the kindergarten through twelfth grade education area, and the versatility of applications using these special devices is particularly suited for educational applications.

The *LinkWay Toolkit*, available from Washington Computer Services, is a special set of routines to be used with *LinkWay* to extend its capabilities. The *Toolkit* contains a number of binary routines that are used as drivers to control special devices or operations. In addition, the *Toolkit* contains a number of extra fonts for the system, and routines for graphics, animation, and speech, with sample *LinkWay* application folders.

The arena of graphic, object oriented applications is a growing one, with several products either available now or on the horizon. *LinkWay* is a first release by IBM into this application area. The program was designed to provide a data organization tool that applies hypertext principles, runs on any PC, provides access to color graphics and other functions of the IBM PC, and is easy to use. Many *HyperCard* applications can be duplicated in function using *LinkWay*, and several developers are working on adaptations of *HyperCard* applications into the IBM world. Several sample applications are provided with *LinkWay* which illustrate some of the possibilities, and it is only a matter of time until we see the same proliferation of applications as are currently available for *HyperCard*.

Product Information

Product Name: *IBM LinkWay*
Company Info:
IBM
P.O. Box 1328-W
Boca Raton, FL 33429
(404) 238-3245
Price: $110
Developer: Larry Kheriaty

Product Name: *LinkWay Toolkit*
Company Info:
Washington Computer Services
2601 North Shore Road
Bellingham, Washington 98226
(206) 734-8248
Price: $300 (Site license $2000)
Developers: Larry Kheriaty

Shafer On Scripting

Overriding HyperCard's *Message-Passing Scheme*

by Dan Shafer

One of the most powerful ideas in *HyperCard* is the message-passing hierarchy. By taking advantage of this capability, you can write HyperTalk scripts and handlers that are as generic as possible and, therefore, reusable. Code reusability is one of the key ideas in the current software engineering rage known as object-oriented programming because of the tremendous gains in efficiency it brings to programmers.

Just so we're sure we're all starting at the same point, let's quickly review the message-passing hierarchy. Then I'll talk a little about reasons and methods for overriding this normal message-passing scheme.

The Norm

Left to its own devices, HyperTalk receives any message at a default location in the hierarchy shown in Figure 1 and passes it up the chain until one of two things happens. Either a handler with the same name as the message is found and the message is interpreted into actions, or the message reaches the top level of the hierarchy. If a system message reaches the top level of the hierarchy, *HyperCard* simply tosses it away. But if a user-defined message gets that far without being handled, an error is generated.

If you're skeptical — or if you just need a concrete demonstration of abstract ideas before you can own them — you can prove the accuracy of this statement by writing two one-line button scripts. The first, shown in Listing 1, generates a system message using the `send` command that we'll spend more time on later in this column. The second (Listing 2) is exactly the same, except this time we've defined a message that is not in the system's vocabulary. Pressing the first button has no visible effect; the message is simply ignored. But pressing the second generates an error message: "Can't understand openMind."

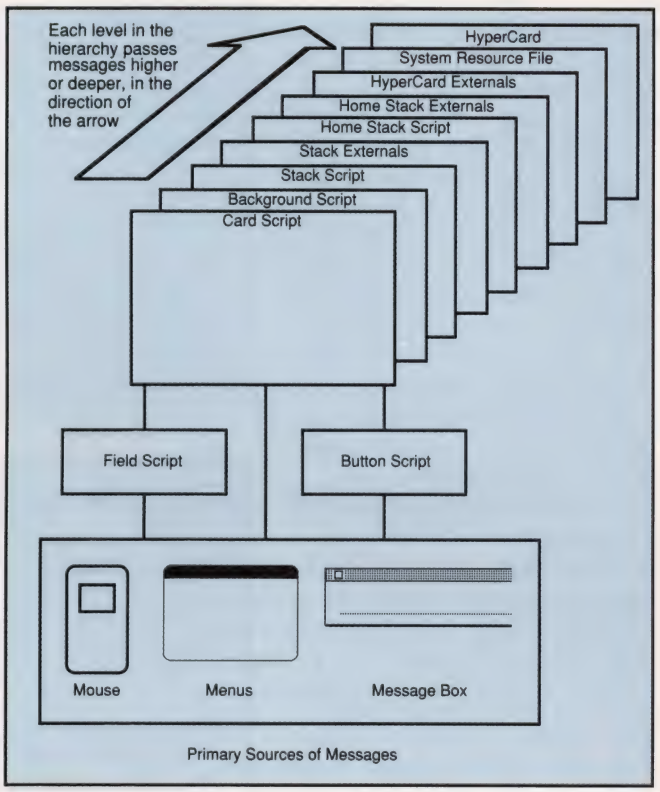


Figure 1

This diagram, taken from Chapter 5 of HyperTalk Programming (including version 1.2) *by the author, demonstrates the basic message passing hierarchy in* HyperCard.

Listing 1

```
on mouseUp
  send "openCard"
end mouseUp
```

Listing 2

```
on mouseUp
  send "openMind"
end mouseUp
```

Message-Eating and Passing

There are occasions on which you wish to override this normally useful message-passing scheme imple-

Dan Shafer is the author of many microcomputer books, including the newly released HyperTalk Programming (including version 1.2), *and* Understanding HyperTalk *(both from Hayden Books).*

mented by *HyperCard*. When you wish to do so, HyperTalk thoughtfully provides two commands to enable you to defeat its standard behavior: `pass` and `send`.

Recall that when a message encounters a handler that bears its name, it executes that handler and then essentially disappears from the message-passing hierarchy. Generally, this is what we want—one message, one handler. But you've probably made the mistake (I know I have!) of intercepting a message with a handler and then experiencing bizarre behavior as a result. For example, the `doMenu` message is often used to disable or confirm certain user actions. Listing 3 shows an example of a typical use of the `doMenu` command.

This handler does exactly what you intend: Confine the user to the current background even when "Next" or "Prev" are chosen from the *HyperCard* "Go" Menu. But if you implement that handler and then try do something unrelated to the "Go Next" or "Go Previous" menu command, you're in for a surprise. None of the other menu items work at all (except, it turns out, tool selection, which bypasses the `doMenu` command in *HyperCard 1.2.2*, I understand this anomalous behavior is changed in a future release). Why? Because you've built a handler to intercept the `doMenu` system message and that handler has taken care of the user's menu choice by determining if it is one it should handle. If it is, it carries out the commands in the handler. If not, it simply "eats" the command.

If you want to leave other menu options enabled, you must add a `pass` command at the end of the handler, as shown in Listing 4. Notice that we simply `pass` the command, not its associated parameters. *HyperCard* takes care of the parameters in a `pass` command as it passes the entire operation string that initiated the message in the first place.

The `pass` command is often helpful in implementing *HyperCard* stacks efficiently. It enables you to define groups of objects, such as buttons, that have some common functionality, but also some unique capabilities or requirements of their own. Listings 5 and 6 represent two hypothetical *HyperCard* buttons whose scripts obviously have some overlapping requirements in the final three lines. With only two buttons, this situation is at least tolerable (though duplicating the functionality is still not good design), but what if you had 10 or 15 such buttons? The resulting wasted stack

Listing 3

```
on doMenu what
  if what is "Prev" then go previous card of this ¬
  background
  if what is "Next" then go next card of this background
end doMenu
```

Listing 4

```
on doMenu what
  if what is "Prev" then go previous card of this ¬
  background
  if what is "Next" then go next card of this background
  pass doMenu
end doMenu
```

Listing 5

```
on mouseUp -- script for hypothetical button 2
  open file "Smalltalk Advantages"
  read from file "Smalltalk Advantages" ¬
  until numToChar(26) -- EOF marker
  put it into field "Language Pluses" of card "Smalltalk"
  put the number of lines of it into totalLines
  play "harpsichord" cs f a
  push this card
  visual effect zoom open slowly to black
  go to card "Smalltalk"
  put "Here are the" && totalLines && ¬
  "advantages of Smalltalk."
  wait until the mouseClick
  pop card
end mouseUp
```

Listing 6

```
on mouseUp -- script for hypothetical button 2
  open file "HyperTalk Advantages"
  read from file "HyperTalk Advantages" until ¬
  numToChar(26) -- EOF marker
  put it into field "Language Pluses" of card "Smalltalk"
  put the number of lines of it into totalLines
  play "harpsichord" cs f a
  push this card
  visual effect zoom open slowly to black
  go to card "HyperTalk"
  put "Here are the" && totalLines ¬
  && "advantages of HyperTalk."
  wait until the mouseClick
  pop card
end mouseUp
```

Listing 7

```
on mouseUp -- revised script for button 1
  global totalLines,language
  open file "Smalltalk Advantages"
  read from file "Smalltalk Advantages" until ¬
  numToChar(26) -- EOF marker
  put it into field "Language Pluses" of card "Smalltalk"
  put the number of lines of it into totalLines
  put "Smalltalk" into language
  pass mouseUp
end mouseUp
```

space for common scripts could obviously be saved by judicious programming. Listings 7 and 8 show these two button scripts reduced to their unique requirements, although Listing 9 shows a script that would appear higher in the *HyperCard* hierarchy (probably at either the card or background level, depending on where the buttons "live") to supply the common function. Notice that we have gone from two handlers of 12 lines each (24 lines of code discounting open and end statements) to two 7-line handlers and one 8-line handler (22 lines of code). This saves only two lines, but the fact that it saves lines at all when consolidating only two similar handlers demonstrates the power of the idea. If we had, say, 15 programming languages to compare this way, we would go from 180 lines to 113, a savings of 67 lines, or 37% in code space. When scripts are limited to 30,000 characters, savings like these can be important.

Beyond code savings, the reusability of the resulting code is immensely improved. The handlers in Listings 5 and 6 can only be reused in the sense that they can be copied, pasted, and modified by editing. The combination of the handlers in Listings 7-9, on the other hand, produce a single handler (Listing 9) that can serve for any subsequent buttons we add for other languages where the same functionality is needed.

Listing 8

```
on mouseUp -- revised script for button 2
  global totalLines,language

  open file "HyperTalk Advantages"
  read from file "HyperTalk Advantages" until ¬
  numToChar(26) -- EOF marker
  put it into field "Language Pluses" of card "HyperTalk"
  put the number of lines of it into totalLines
  put "HyperTalk" into language

  pass mouseUp
end mouseUp
```

As is often the case, HyperTalk provides a number of ways of accomplishing this consolidation of common functions. The `pass` command is one way. You could also define a message called something like `goCard` and put it higher in the hierarchy, then call it from the `mouseUp` handlers in Listings 7 and 8 rather than passing the `mouseUp`.

You Send Me

If the `pass` command allows us to gain considerable efficiency and reusability of our HyperTalk code, the `send` command adds an order of magnitude of power to the concept. The `pass` command has two inherent limitations. First, it can only `pass` the currently executing command. Second, it can only `pass` that command to the next level of

Listing 9

```
on mouseUp -- script for card or background level
  global totalLines,language
  play "harpsichord" cs f a
  push this card
  visual effect zoom open slowly to black
  go to card language
  put "Here are the" && totalLines && "advantages of"¬
  && language & "."
  wait until the mouseClick
  pop card
end mouseUp -- for card or background level
```

Listing 10

```
on mouseUp
  -- This is the button script
  global score
  ask "What is the student's final score?"
  if it is empty then exit mouseUp
  put it into score
  playMusic -- first handler below is in button script
end mouseUp

on playMusic
  global score
  if score <70 then
    play "boing"
    exit playMusic
  end if
  if score <85 then
    play "boing"
    send playMusic to card 1
    exit playMusic
  end if
  if score < 100 then
    send playMusic to this stack
    exit playMusic
  end if
  -- otherwise,...
  send playMusic to card 1
  send playMusic to this stack
  answer "Perfect Score!!!"
end playMusic
```

Listing 11

```
on playMusic
  -- script at card level (card 1)
  play harpsichord es b f
end playMusic
```

Listing 12

```
on playMusic
  -- script at stack level
  play harpsichord cs d e f g a b
end playMusic
```

the pre-defined *HyperCard* hierarchy.

Sometimes, we want or need to send some message other than the current one to an object at the same level of the hierarchy or lower. When that is the case, we can use HyperTalk's send command. The send command lets us transmit any message—system or user-defined—with or without parameters to any other specifically addressable object. Its power is immense.

You should be careful in using send because it can often become a way of building and excusing poorly designed code. The send command should only be used when it makes the code smaller, easier to follow, and more straightforward to reuse.

Two circumstances that warrant consideration of the use of the send command are:

- When you want to bypass intermediate handlers between the current object and the desired handler
- When an object at this or a lower level of the hierarchy has exactly the code you want to execute and you want to avoid duplicating it

Listings 10, 11, and 12 demonstrate a trivial but illustrative example of the first situation. Here, we have three different sound effects, one at the button level, one at the card level, and one at the stack level. Sometimes, we want to play all of the sounds. Other times, we want to play only one or two of them. The button handler in Listing 10 takes care of the situation nicely.

In my calendar (I use *Organizer+* from Dazzl Software for my schedule), I have a number of handlers that perform special functions such as looking for information that should be exported. When one of these special handlers finishes executing, I want to go back to today's schedule card. I could use a series of push and pop statements to keep track of card travel, but I find that messy, particularly if I'm doing a lot of jumping around in the stack. But the stack includes a button labeled "Today" that returns me to today's card. This button must do some intelligent maneuvering because it can't just simply look for today's date. (The date may appear in a number of places in the stack, for example, in my To Do list.) Rather than repeating this code everywhere I want to go back to today's date, I simply use the command:

```
send mouseUp to background ¬
button "Today"
```

and let the button script take care of the problem for me.

Interfacing The Future

Creating Easy to Use Control Devices in HyperCard

by Craig Ragland

The Mac's graphic interface and flexible pointing device (the mouse) have led many software developers to explore mouse-driven controls and dials. We technological humans are accustomed to controlling machines through direct manipulation of switches, knobs, and sliding controls, so these simulated or virtual controls feel quite natural. If a screen-based control is graphically familiar and behaves in the same manner as its real-world counterpart, it is trivially easy to learn and use, even for naive computer users.

HyperCard and *SuperCard* both offer excellent environments for exploring these types of user interfaces. Although the performance of interpreted HyperTalk on slower Macs may be inadequate for some controls, it suffices for many and is certainly acceptable for prototyping. In this column, we shall explore three types of controls: toggles, knobs, and sliders. None of these are directly available in the standard *HyperCard* or SuperCard interface (except in the simplified case of the standard Apple check boxes and radio buttons). All three of these require the tight integration of graphic objects and clever scripting.

Toggles

The standard Apple check boxes and radio buttons are graphically simple examples of toggles or switches. According to Apple's guidelines, Check Boxes are intended to be used to let users choose among alternative options. Check boxes toggle between two states: on and off. If the box is checked, the option is on; otherwise it is off. Check boxes should also function independently from each other—in groups of check boxes clicking on one has no effect on the others.

HyperCard developers not understanding the check box interface guidelines (or choosing to ignore them) have used check boxes for menu selections, close boxes, navigational links, and to initiate operations (sorts, finds, printing, etc.). In general, if an action begins immediately following a click on a check box button, other than making more options visible, it is inconsistent with the Apple guidelines.

If a screen-based control is graphically familiar and behaves in the same manner as its real-world counterpart, it is trivially easy to learn and use, even for naive computer users.

As pointed out in last issue's "Interfacing the Future" column, according to Apple guidelines radio buttons are intended to be mutually exclusive—only one in a group is on and clicking on any other radio button in a group turns the other one off. [See *HyperLink Magazine* Vol. 2, No. 2—Ed.] This guideline has likewise been ignored by numerous *HyperCard* developers. The small size and strong visual/tactile appeal of a highlighted radio buttons have resulted in their widespread use for a number of different functions. for example, one interesting user interface guideline "violation" used radio buttons to mark towns on a hierarchical set of maps.

Figure 1
Behavior of normal check boxes and radio buttons when the user mouses down on the button. This is not true of HyperCard *buttons (except those in* HyperCard's *dialog boxes).*

HyperCard check box and radio buttons behave slightly differently from those in other applications (or even in *HyperCard's* own dialog boxes). In most applications, mousing down on either a check box or radio button emboldens that button while the mouse is held

Craig Ragland is a principal with INTERACTIVE DESIGN, a Seattle-based Hypermedia development company and can be reached at 206-542-9000. INTERACTIVE DESIGN created 101 Scripts & Buttons for HyperCard *a collection of* HyperCard *interface extensions published by Individual Software: 800-622-9986.*

down (see Figure 1). This provides user feedback on the user's choice before the action has been completed.

The disadvantage of using these standard interface items in ways which are different from the Apple guidelines are really only pertinent when Mac-literate users will be using your stacks. If you design stacks for naive end users, there are no particular reasons you should follow the Apple guidelines. There is nothing embedded in the graphic image of radio buttons or check boxes that suggest how they should function when they occur in groups. The guidelines for the behaviors of these interface items, like many others are really rather arbitrary conventions which add to the consistency of Macintosh software as a whole. If your stacks will be used by Mac-naive users you have fewer constraints over the interfaces which you design.

Extending *HyperCard* Buttons

More complex (and visually interesting) toggles or switches can be created using graphic objects along with associated scripts. In *HyperCard* the graphic objects can be created as ICONs or FONT characters. *SuperCard* allows even greater flexibility because paint frames and draw-type graphics can be independently manipulated. Figure 2 shows several toggle switches from Individual Software's product *101 Scripts & Buttons for HyperCard* (created by the author). The scripts for these buttons use an ICON flipping animation technique to simulate the real world version of the object. The script for toggle button #1 is in Listing 1 and the script for toggle button #7 can be found in Listing 2.

There is a very wide range of other toggles and switches. It is not terribly difficult to simulate almost any form of two phase switch (some examples you might try include bat switches, light switches, sliding toggles, etc.) An extensive and spectacular collection is found in "StackStarter" by Robertson Smith.

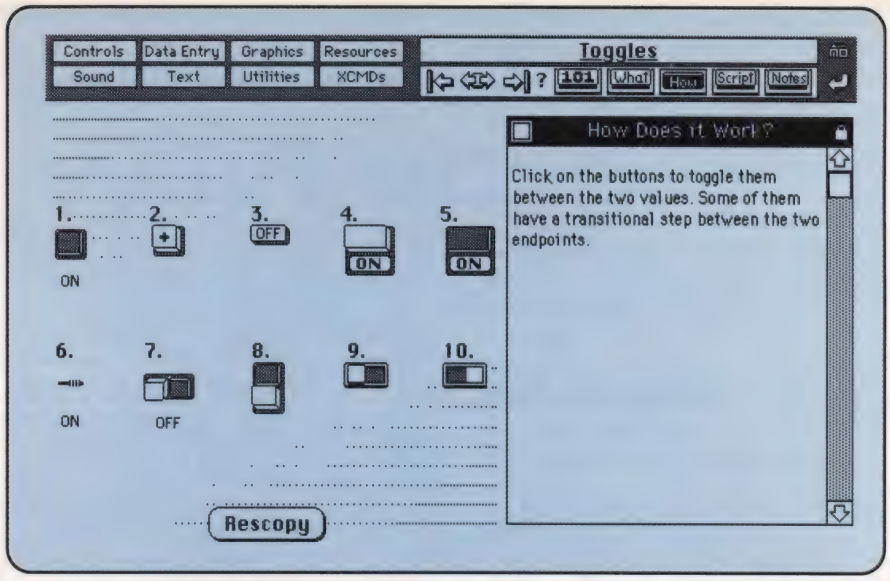


Figure 2

This is the card devoted to Toggles from 101 Scripts & Buttons for HyperCard *showing several different ICON based toggle switches. The script for toggle button #1 is in Listing 1, and the script for toggle button #7 is in Listing 2.*

Listing 1 - Script for Toggle #1 in Figure #2

```
on mouseDown
  --  ICON Resources used: ID# 18752: "SquarePlain",
  --                       ID# 16918: "SquareGray"
  if icon of me = "18752" then
    set icon of me to "SquareGray"
    play boing c2s
    set name of me to "ON"
    --  Put code for what happens when button is "Gray"
    -- or "ON" here.
  else
    set icon of me to "SquarePlain"
    play boing e2s
    set name of me to "OFF"
    --  Put code for what happens when button is
    -- "White" or "OFF" here.
  end if
end mouseDown
```

Listing 2 - Script for Toggle #7 in Figure #2

```
on mouseDown
  --  ICON Resources used: ID# 6237:  "RWhiteS"
  --                       ID# 10677: "LWhiteS"
  if the mouseH < item 1 of loc of me then
    set icon of me to "RWhiteS"
    set name of me to "ON"
    -- Put code for what happens when switch
    -- is turned "ON" here.
  else
    set icon of me to "LWhiteS"
    set name of me to "OFF"
    --   Put code for what happens when switch
    --  is turned "OFF" here.
  end if
end mouseDown

on mouseStillDown
  if the mouseLoc is within rect of me then send ¬
  mouseDown to me
end mouseStillDown
```

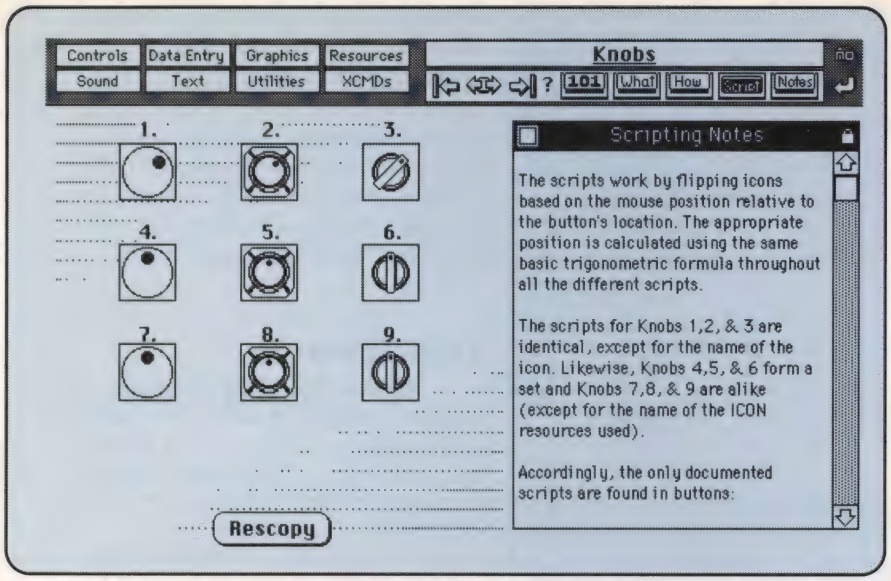


Figure 3

This is the card devoted to Knobs from 101 Scripts & Buttons for HyperCard *showing several different ICON based Knobs.*

Listing 3 - Script for Knob #1 in Figure #3

```
on mouseDown
  send mouseStillDown to me
end mouseDown

on mouseStilldown
  if the mouseLoc is within rect of me then
    -- Create list of ICONs to use, since a discontinuous
    -- subset of all the "Dial." ICONs are used.
    --  This button uses the following ICON resources:
    --  ID# 11962: "Dial.2",  ID# 834:   "Dial.4"
    --  ID# 30117: "Dial.6",  ID# 4264:  "Dial.8"
    put "2,4,6,8" into iconList
    --  Compute difference between mouseLoc and button
    put (item 1 of the mouseloc+1) - ¬
    (item 1 of loc of me) into deltaX
    put (item 2 of loc of me) - ¬
    (item 2 of the mouseloc+1) into deltaY
    -- deltaY/deltaX = Tangent of angle from
    -- knob loc to mouseloc.
    put deltaY/deltaX into t
    put atan (t) * 57.29578 into d
    -- Convert differences into degrees, then determine
    -- the appropriate position number (posNumber).
    if deltaX < 0 then add 180 to d
    if d ≤ 0 then add 360 to d
    put trunc(((405+45 - d) mod 360)/90)+1 into posNumber
    --  Little fix to avoid rounding errors.
    if posNumber < 1 or posNumber > 4
    then put 1 into posNumber
    put "Dial."& item posNumber of iconList into iconName
    set the icon of me to iconName
    --  Additional code to determine what button does
    if posNumber = "1" then
    --  Put code here for position 1 choice.
     else if posNumber = "2" then
       --  Put code here for position 2 choice.
    else if posNumber = "3" then
       --  Put code here for position 3 choice.
    else
       --  Put code here for position 4 choice.
    end if
  end if
end mousestilldown
```

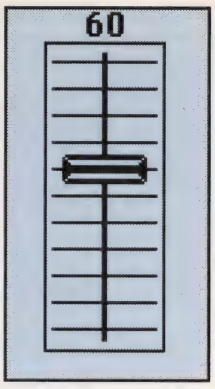


Figure 4

A sliding control from 101 Scripts & Buttons for HyperCard. *The script for the icon is found in Listing 4, but the script for the transparent button overlaying the paint graphics is found in Listing 5.*

Knobs

Although toggle buttons are great for specifying one of two states, they are not terribly good for indicating a single option when there are multiple possible states. Turnable knobs are an excellent way to input a single condition from several options. This seems particularly appropriate for variables which bear similarities to those controlled by knobs in the real world, e.g., volume or temperature settings.

Listing #3 shows a fairly general solution to flipping ICONs for the simulation of a turning knob. It is quite easy to make minor changes to this basic script for knobs with different numbers of discrete positions or different angular positions. There are also many other scripting approaches which can be used to simulate turning knobs.

Although knobs allow access to more values than toggles, they are constrained by the number of discrete positions in a circle which you can reproduce on a graphical screen of limited resolution. The practical limit for *HyperCard* seems to be about 16 positions. While you can create icons for more than 16 positions, they are very difficult to visually distinguish. Sliding controls offer another alternative with many more potential settings.

Sliders

Sliding controls, or sliders, typically move a pointer or indicator along a straight-line axis. The most familiar sliding control on the Mac is the scroll bar—but don't forget the volume control on the Control Panel Desk Accessory. With sliding controls the range of alternatives is only limited by the number of pixels on the axis (100 in the case of Figure

Listing 4 - Script for Slider Icon in Figure 4

```
on mouseDown
  put loc of me into newLoc
  put item 1 of newLoc into XHere
  repeat while the mouse is down
    put the mouseV into item 2 of newLoc
    if item 2 of newLoc ≥ 82 and ¬
    item 2 of newLoc ≤ 182 then
      set loc of me to newLoc
      put 100 - (item 2 of newLoc - 82) into lineNum
      put lineNum into cd fld Value
    else
      if item 2 of newLoc < 82 then set loc of me to¬
      XHere,82
      else set loc of me to XHere,182
      put 100 - (item 2 of loc of me - 82) into lineNum
      put lineNum into card field Value
    end if
  end repeat
  if item 2 of the loc of me ≥ 82 and item 2 of the ¬
  loc of me ≤ 182 then
    put 100 - (item 2 of loc of me - 82) into lineNum
  else
    if item 2 of the loc of me < 82 then set loc of me ¬
    to XHere,82
    else set the loc of me to XHere,182
    put 100 - (item 2 of loc of me - 82) into lineNum
  end if
  put lineNum into cd fld Value
end mouseDown
```

Listing 5 - Script for Transparent Button over Scale in Figure 4

```
on mouseDown
  set loc of cd btn "Pointer" to item 1 of loc of cd btn ¬
  "Pointer",the mouseV
  send mouseDown to cd btn "Pointer"
end mouseDown
```

4). It is possible to use any arbitrary angle, or even curved lines, as the axis, but it is much simpler to program using the X or Y axis. Listings 4 and 5 provide scripts for controlling a vertical sliding controller at a specific screen position. The Listing 4 script could be improved by making it independent of screen location.

Conclusions

All three of these graphical controls (toggles, knobs, and sliders) offer richer methods of data input than text entry. They share several advantages over text entry:

- Due to their real world counterparts they are equally familiar to both experienced and naive computer users.
- It is impossible to enter completely invalid data, thus eliminating the need for data validation.
- They offer continuous visual feedback during the process of indicating the user's choice.
- Because they are mouse-driven, they do not require knowledge of typing or even the use of a keyboard.

Short Stacks

"Securities Grapher" Is a Stack That Makes Practical Use of the Paint Tools

by William K. Balthrop

The Paint tools in *HyperCard* are a versatile set of graphic devices. Used under script control they can create attractive and utilitarian charts and graphs. To demonstrate the techniques involved, we've devised a "Securities Grapher" stack for this issue's "Short Stack." This stack presents a simple but effective approach to converting a table of data for a stock into a "HiLo" chart (see Figure 1), which should be familiar to anyone who has ever tracked securities. This stack tracks a single security for 116 days, and it automatically scales the values that have been input so the highest value appears at the top of the chart, the lowest at the bottom, with the others correctly scaled in between.

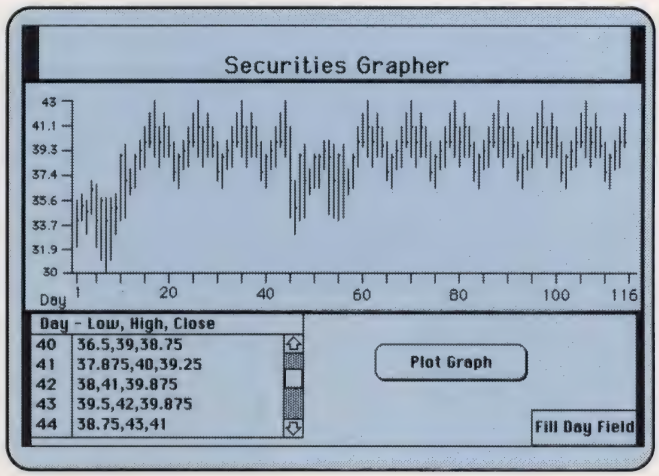


Figure 1

This is the "Securities Grapher" stack's only card. This "HiLo" chart helps demonstrate some scripting techniques for using the Paint tools to translate data into a more easily understood form.

Creating the Graph

To create the screen in Figure 1, ensure that you are in scripting level by checking the "Scripting" check box on the last card of your "Home" stack. Next, select "New Stack..." from the File menu. Name the stack "Securities Grapher" and be sure that the "Copy Current Background" option is *not* checked.

Because the "Securities Grapher" paints to specific locations on the screen it is critical that the graph is set to exactly the correct locations on the screen. The "Home" script of the latest version of *HyperCard* (version 1.2.2) contains an excellent handler called `xy` that can aid in this process.

```
on xy
-- puts the mouse location in the message box
-- until a mouse click
  set the cursor to cross
  repeat until the mouse is down
    put the mouseLoc
  end repeat
  set the cursor to browse
end xy
```

If you haven't updated your "Home" stack by putting the latest scripts into it, be sure to add this one to your "Home" stack script before you go any further.

Anytime you want to get an exact screen location, just show the Message box, type "xy" into it, and press Return. As you move the mouse, the horizontal and vertical positions of the mouse are displayed until you click the mouse again.

Let's begin by creating the vertical axis of the graph. Tear off the Tools menu and place it near the bottom-right corner of the screen. Double click the Line tool and select the narrowest line, the one on the far left. Select the Line tool from the Tools menu, show the Message box, type "xy," and press Return. Now, without clicking, move the mouse toward the upper-left corner of the screen until the Message box reads "40,62." Next, draw the vertical axis by holding down the mouse button and dragging straight down to a point about two thirds of the way down the screen. Again, type "xy" followed by Return into the Message box to check where the bottom end of the line is. You want it to be at screen location "40,202." Adjust the line by either extending it with the Line tool or erasing it with the Erasure tool until the line goes from precisely "40,62" at the top to "40,202" at the bottom.

Next, using the Line tool again, draw the horizontal axis from "40,202" to "506,"202." Now add the horizontal and vertical tick marks as shown in Figure 1.

William "Kelly" Balthrop is the Director of R & D at HyperLink *and has been developing and publishing computer software for the last decade.*

These marks are 20 pixels apart and 8 pixels long—i.e., the marks on the vertical axis are placed at 62, 82,102,122, etc., and the horizontal axis marks are placed at 40, 60, 80, 100, 120, 140, and so on.

The stack's title, and the day numbers across the bottom are painted on using the Text tool (the large A) on the Tools menu. We used 12 point Geneva and put a label every 20 days.

Adding the Fields

Down the left side of the graph there are eight fields that hold the range of values of the securities. One of the nicest parts of "Securities Grapher" is that it automatically scales the values and inserts appropriate numbers into these fields. All eight fields are identical except for their names, which must be "Scale 1," "Scale 2," "Scale 3," and so on, for the scripts to work properly. The bottom field (containing the number 30 in Figure 1) is named "Scale 1" and the top field (containing 43 in Figure 1) is named "Scale 8."

To create these fields, first select "New Field" from the Objects menu, then "Field Info..." from the Objects Menu, and name the field "Scale 1" by entering the name in the "Field Info..." dialog box.Select the font by clicking on the "Font" button in the box, and set the font to 9 point Geneva. After you click "OK," move and resize the field so it fits in between the bottom tick mark on the vertical axis and the left edge of the *HyperCard* screen. Now you can "clone" the remaining fields by holding down the Option key, and clicking and dragging up on the first field. Each time you create the new field, be sure to rename by opening its "Field Info..." as we did above.

The Data Fields

The fields at the bottom left of the screen are where the data is entered for the graph. These two fields appear to be a rectangular field on the left and a scrolling field on the right, but actually they are both scrolling fields with the field on the right on top of the left one, covering its scroll bar. You create these by again selecting "New Field," and then "Field Info..." from the Objects menu. Next name the field "Day,"

Listing 1 - Script for Card Field "Daily Data"

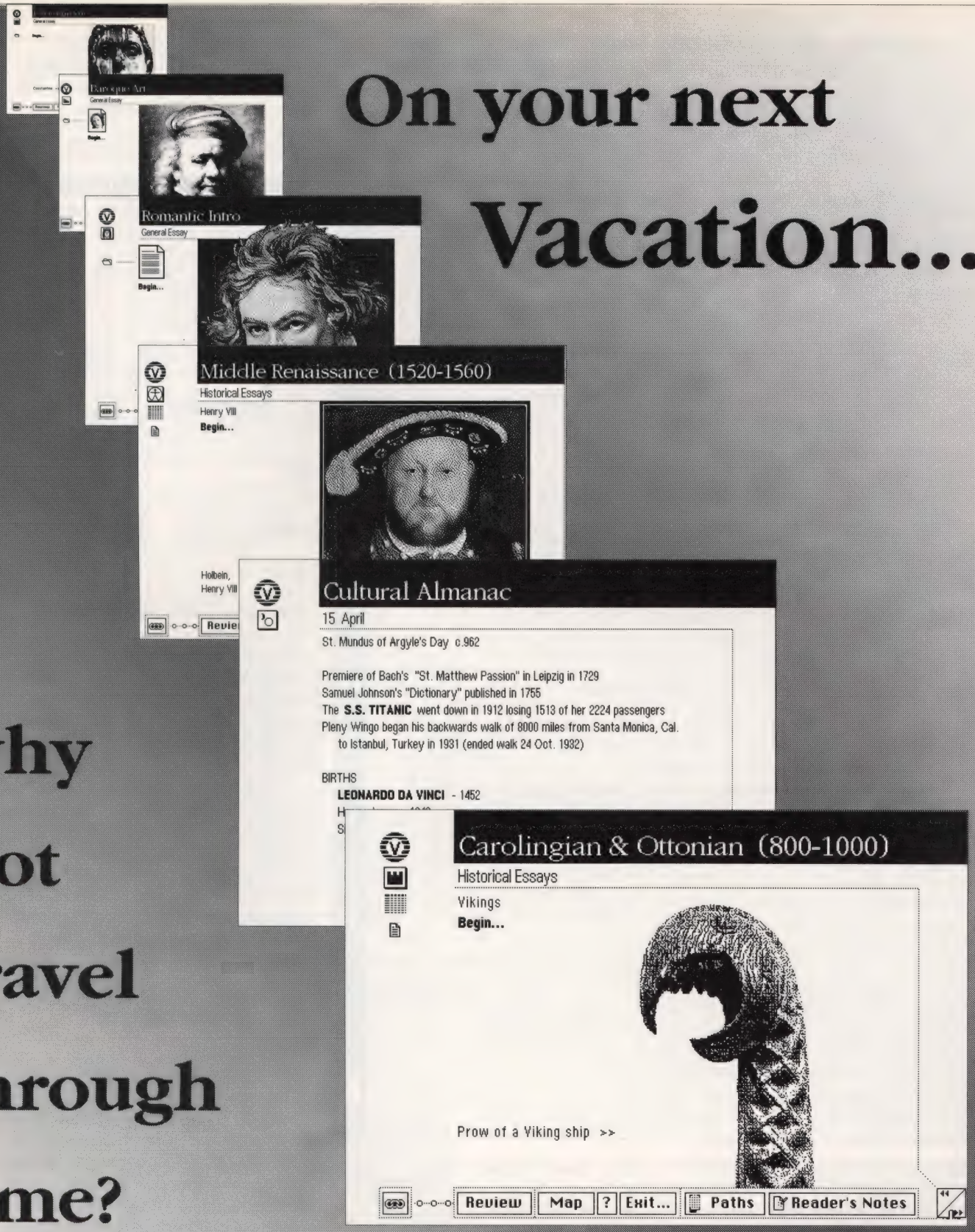
```
on mousewithin
  set the scroll of card field "Day" to ¬
  the scroll of card field "Daily Data"
end mouseWithin
```

Listing 2 - Script for Card Button "Fill Day Field"

```
on mouseUp
  set cursor to busy
  repeat with num=1 to 116
    put num into item 1 of line num of cd fld "Day"
    set cursor to busy
  end repeat
end mouseUp
```

Listing 3 - Script for the "Plot Graph" Button

```
on mouseUp
  global mn,mx
  put number of lines in cd fld "Daily Data" into numDays
  put 9999 into mn
  put 0 into mx
  set cursor to busy
  put "Please wait while I evaluate the data. . ."
  If numDays > 116
  then
    put "116 days is the maximum for this stack!"
    exit mouseUp
  end if
  repeat with lineNum =1 to numDays
    set cursor to busy
    if line LineNum of cd fld "Daily Data" is empty
    then exit repeat
    put checkData(line lineNum of cd fld "Daily Data") ¬
    into minMax
    if word 1 of minMax ="Error" then
      put lineNum into word 4 of minMax
      put minMax
      exit mouseUp
    end if
    put item 1 of minMax into mn
    put item 2 of minMax into mx
  end repeat
  put lineNum into NumDays
  clearGraph
  set numberformat to "0.#"
  get (mx - mn) / 7
  repeat with scale = 1 to 8
    set cursor to busy
    put Mn +it * (scale-1) into cd fld ("Scale" && scale)
  end repeat
  put 140 / (Mx - Mn) into adjuster
  choose line tool
  set lineSize to 1
  repeat with lineNum=1 to numDays
    set cursor to busy
    put line lineNum of cd fld "Daily Data" into theData
    put item 1 of theData into Low
    put item 2 of theData into High
    put item 3 of theData into Close
    put trunc(44+((lineNum - 1)*4)) into item 1 of Start
    put trunc(202-(Low Mn)*adjuster) into item 2 of Start
    put trunc(44+((lineNum-1)*4)) into item 1 of Stop
    put trunc(202-(High-Mn)*adjuster) into item 2 of Stop
    put min(item 2 of Start,201) into item 2 of Start
    put max(item 2 of Start,62) into item 2 of Start
    put min(item 2 of Stop,201) into item 2 of Stop
    put max(item 2 of Stop,62) into item 2 of Stop
    drag from Start to Stop
    click at trunc(45 + ((lineNum - 1) * 4)),trunc(202 - ¬
    (Close - Mn) * adjuster)
```



On your next Vacation...

why not travel through time?

Cultural Resources, Inc
Creative Software for the Home, Educational Institutions and Libraries
7 Little Falls Way, Scotch Plains, NJ, 07076

with HyperCard™ and Culture™ 1.0

$175.00
7 disk set

To Order Call
201-232-4333

HyperCard™ is a trademark of Apple Computer, Inc.

Listing 3 - Script for the "Plot Graph" Button - continued

```
  end repeat
  choose browse tool
  unlock screen with visual dissolve
  put "Graph complete!"
  wait 3 seconds
  put empty
  hide msg
end mouseUp

function CheckData latestData
  global mn,mx
  repeat with x=1 to the number of items of latestdata
    put CheckNum(item x of latestdata) into Cond
    if Cond is "Error" then
      put "Error in day x - illegal number" into minMax
      return minMax
    end if
  end repeat
  if the number of items in latestData ≠ 3 then
    put "Error in day x – wrong number of items!" into ¬
    minMax
  else if min(latestData) ≠ item 1 of latestData then
    put "Error in day x – item 1 not Lowest!" into minMax
  else if max(latestData) ≠ item 2 of latestData then
    put "Error in day x – item 2 not Highest!" into minMax
  else put min(min(latestData),mn) & "," & ¬
 max(max(latestData),mx) into minMax
  return minMax
end checkData

function CheckNum Num
  put 0 into decimals
  repeat with curnum = 1 to length(Num)
    set cursor to busy
    if char curnum of num = "." then add 1 to decimals
    if char curnum of num is in "1234567890." and¬
    decimals < 2 then next repeat
    else return "Error"
  end repeat
  if Num = "." then return "Error"
  if Num = empty or Num < 0 then put 0 into num
  add zero to Num
  return Num
end CheckNum

on ClearGraph
  lock screen
  choose select tool
  drag from 44,201 to 508,62
  domenu "Cut Picture"
  choose browse tool
  unlock screen
  set the cursor to watch
  lock screen
end ClearGraph
```

and select "Scrolling" under "Style," then press the "Font" button, and choose Chicago, 12 point. Now, move and resize the field to go in the lower-left corner of the screen, so it will hold 5 lines of text, and be wide enough to easily hold 3 characters across. Next, clone the left-hand field by holding down the Option key, then click and drag to the right on the "Day" field. Remember to leave this new field so it covers the scroll bar of the "Day" field. Choose "Field Info..." from the Objects menu and name this new field "Daily Data." Click "OK" and resize this field so it looks like the field in Figure 1.

Now, enter the script in Listing 1 into the "Daily Data" field's script. This script synchronizes the scrolling of these two fields. There are drawbacks to this relatively simple approach, but as long as you keep the mouse cursor within the "Daily Data," the two fields remain synchronized.

Creating the Buttons

Now we're ready to create the three buttons on the card:

- "Day - Low, High, Close"
- "Fill Day Field"
- "Plot Graph"

The "Day - Low, High, Close" button is a rectangular button that acts as a label for the "Day" and "Daily Data" fields we just created.You could just as easily use paint text or a field, but creating a rectangular button with its "Show Name" checked is effective for this purpose.

The "Fill Day Field" button simply fills the "Day" field with the number 1 through 116, so it will allow us to easily identify the data in the "Daily Data" field. Once you create this button, enter the script shown in Listing 2, and click on it once, you can delete it. After that is a good time to set the `lockText` of the "Day" field to true in its "Field Info..." dialog box.

The real workhorse of this stack is the "Plot Graph" button. Just choose "New Button" from the Objects menu, and then select "Button Info..." and name the button "Plot Graph" in the dialog box. Next click the "Script" button and enter Listing 3 into the button's script window.

Using "Securities Grapher"

To use this stack, all you need to know is how the data in the "Daily Data" field is to be formatted. Each line must have 3 numbers separated by commas. The first number must be the lowest, the middle number the highest, and the third number must be either in between these two or equal to one of them. Each line must be ended by pressing Return—if you just press the space bar until you reach the next line, you will get an error when you press "Plot Graph." Have fun, and good graphing!

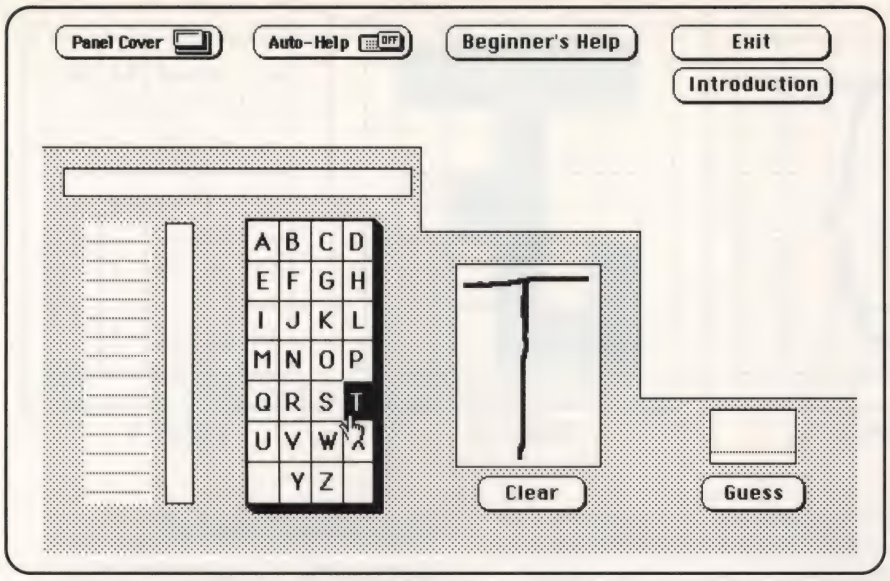


Figure 3

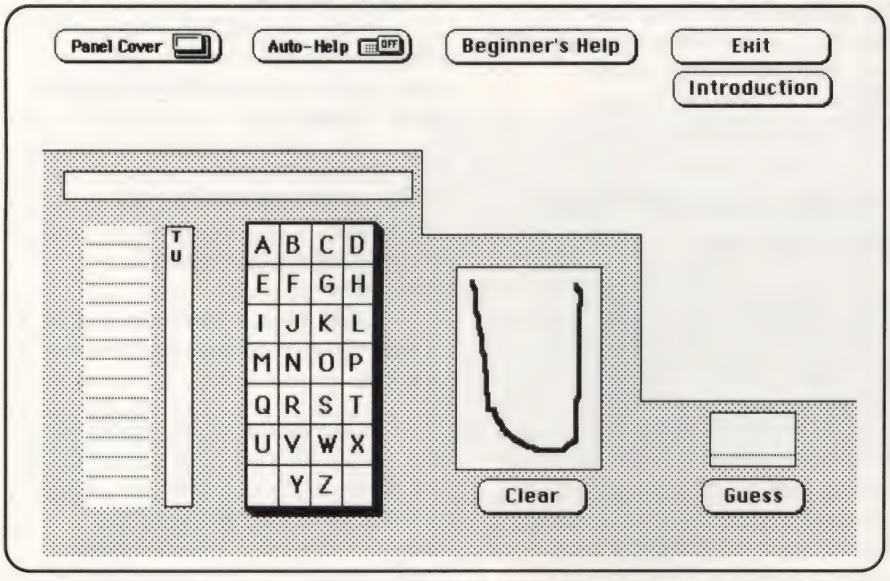


Figure 4

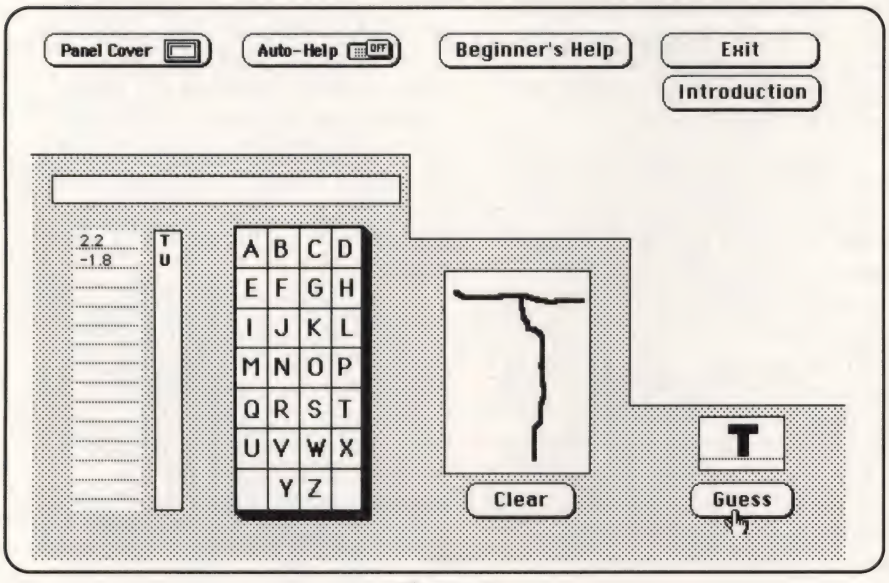


Figure 5

The "Letter Learner Stack"

—continued from page 14

Perhaps one day a neural network will be built that can examine a stock and decide which category it falls in: "Buy" or "Don't buy."

It should be pointed out that conventional artificial intelligence has grappled with all of these problems, but has generally failed to solve them. Neural networks are an alternative to conventional AI.

By the way, the "Letter Learner" stack contains a list of current references on neural networks.

How to Use "Letter Learner"

"Letter Learner" is the first in a series of NeuralStacks—stacks that demonstrate and teach neural networks. "Letter Learner" can be taught to recognize letters of the alphabet that you draw using the mouse.

When we begin, it knows nothing. It will not recognize any letters because you haven't taught it any yet. We can teach "Letter Learner" to recognize letters by drawing a letter into the box in the middle of the screen, and then pressing the button corresponding to that letter (see Figure 3)

Now "Letter Learner" has learned the letter T. The letters that it learns appear in the column on the far left of the screen. If we use the same method to teach the letter U, the screen appears as shown in Figure 4.

Now that "Letter Learner" knows some letters, let's test its knowledge. We can draw any one of the letters it knows, and press the "GUESS" button in the lower right-hand corner. "Letter Learner" guesses T correctly (see Figure 5).

The numbers at the far left can range from –5 to +5, and they represent the amount of "U-ness" and "T-ness" of the letter you just drew. The letter with the highest value is the letter it guesses. "Letter Learner" can learn up to 13 letters at a time.

Now that we've seen how to use "Letter Learner", let's examine how "Letter Learner" works.

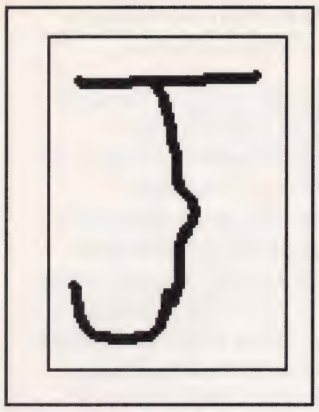


Figure 6

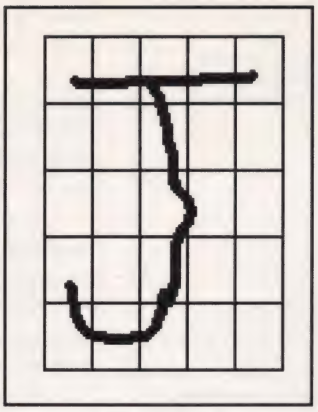


Figure 7

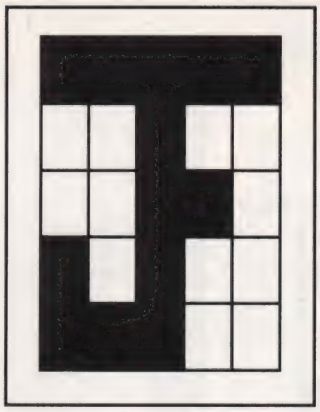


Figure 8

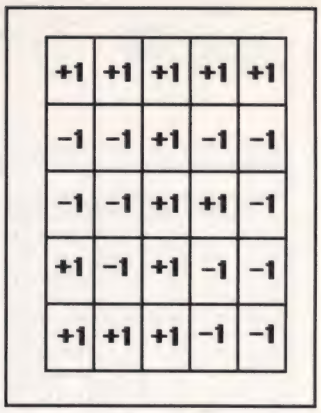


Figure 9

How "Letter Learner" Works

When a letter is drawn into "Letter Learner", as in Figure 6, "Letter Learner" views it atop a 5 by 5 grid (see Figure 7).

For each square in the grid, "Letter Learner" determines whether or not anything has been drawn in that rectangle (see Figure 8).

If something has been drawn in that rectangle, the rectangle is assigned a value of +1. If nothing has been drawn in the rectangle, that rectangle is assigned a value of –1 as shown in Figure 9.

This series of numbers is then read left-to-right, top-to-bottom, creating what is called the "input vector." When you press the "J" button, a single neurode is allocated to represent the letter, and the input vector is fed as input to that neurode.

The J neurode "remembers" that series of numbers in a form called the "weight vector," and when guessing a letter, it compares the input vector for the letter you've drawn to its weight vector. If you teach more than one J to "Letter Learner", rather than allocate two neurodes for that letter, "Letter Learner" simply modifies the weight vector for the J neurode to reflect all the J's it has learned. This is explained in detail in the Help section of the "Letter Learner" stack.

There are many different types of neural networks, and all of them use variations of the concepts we've just discussed. Most neural networks, however, don't use a strategy of one neurode per concept. Most neural networks connect the neurodes together in one pattern or another, and it's the combination of neurodes that fire that determine which concept is being represented. The method that "Letter Learner" uses, that of one letter per neurode, is a variation on MADALINE, invented in 1960 by Bernard Widrow at Stanford University.

"Letter Learner" and *HyperCard*

"Letter Learner" is a *HyperCard* stack, and its safe to say that without *HyperCard*, I'd still be just starting the project. After using *HyperCard*, I can't conceive of ever using a more traditional programming language for a complete project again. Nonetheless, writing "Letter Learner" was a far from trivial task. At least one function that my program needed to perform could not be done with straight HyperTalk commands, and five of the mathematical functions that I needed to process the input vector and the weight vectors were simply too slow in HyperTalk, so I turned to *Turbo Pascal*.

As you may know, HyperTalk allows us to write procedures and functions in other languages, such as Pascal or C, and call these procedures from a HyperTalk script. These procedures and functions are called XCMD's and XFCN's, respectively.To take care of the problems I was having with HyperTalk, I wrote six XCMD's and XFCN's in *Turbo Pascal*. We'll discuss these XCMD's in a moment, but first, let's examine how the concepts of neurodes, input vectors, and weight vectors have been implemented in "Letter Learner."

Mapping Neural Network Concepts to *HyperCard*

In the MADALINE, there are only four neural network objects that must be represented in *HyperCard*. These are:

- The input vector
- The weight vector for each neurode
- The output value for each neurode
- The delta vector, which is sort of a scaled-down version of the input vector

Mapping these into *HyperCard* is not very difficult. The input vector is a series of 25 numbers that represent the letter that the user has drawn in the box. This has been implemented in "Letter Learner" with a single field, appropriately called "InputVector." The 25 numbers in the input vector are represented by 25 items in the field. The output value for each neurode is a single number that represents a comparison between the input vector and a neurode's weight vector, and these are stored in a field called "output" that appears on the far left of the screen.

Representing the weight vector for each neurode is only slightly more complex. As previously discussed, for every neurode there is a weight vector, a series of 25 numbers, that represent that neurode's memory of the letter that it is responsible for. Rather than have a separate field for each neurode, "Letter Learner" uses a single field that holds the weight vectors for all the

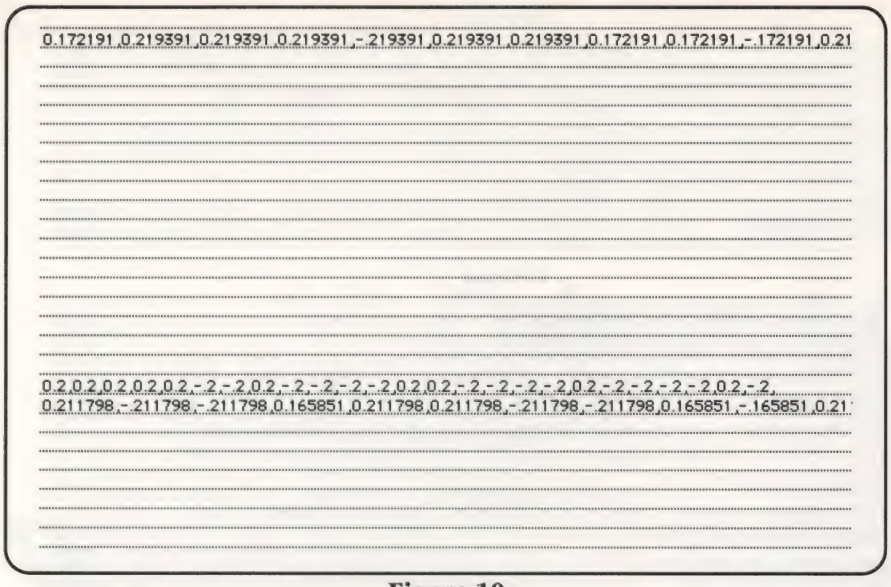


Figure 10

In the field "WEIGHTS" after having learned the letters B, T, and U. Since B, T, and U, the 2nd, 20th, and 21st letters of the alphabet, only those lines of the field have values in them. The weight vectors for B and U are too long to fit on the screen.

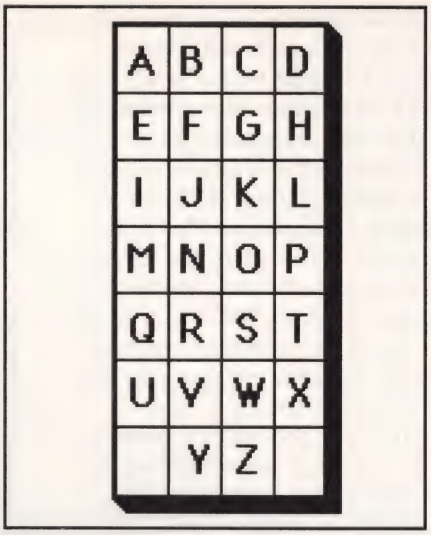


Figure 11

A single button covers this section of the screen. A somewhat complicated formula is used to figure out what section of the button the user pressed, and hence, what letter is underneath that section.

neurodes. Since there are only 26 letters of the alphabet, and since "Letter Learner" only allocates one neurode per letter, we need at most 26 different weight vectors. The field "Weights" holds all the weight vectors for all the letters that have been learned. The weight vector for the A neurode is stored in line 1 of this field; the weight vector for the E neurode is stored in line 5 of this field, etc. Again, each line consists of the 25 numbers in that vector, separated by commas. This allows us to refer to the 7th element of the weight vector for the E neurode as

```
item 7 of line 5 of field ¬
"Weights"
```

The field "Weights" is shown in Figure 10, and it represents the state of the network after having learned the letters B, T, and U.

It should be pointed out that "Weights" is a hidden field, and in addition, it has been set to a 1 pixel by 1 pixel rectangle. Thus, inside "Letter Learner," there is no way to examine this field directly. Instead, "Letter Learner" allows you to examine a single line of the field by turning on a switch called "Show Vectors." When this switch is on, the weight vector for the letter you've just taught "Letter Learner" appears in a field on the bottom of the screen. In addition, "Letter Learner" provides a 5 by 5 grid that allows you to examine a graphical representation of the weight vector for any neurode you like (see the stack itself for more details).

Listing 1

```
if allocated(letterNum) and¬
  ((card field "numNeurodes") <= maxNeurodes) then
    ProcessNeurode(letterNum)
  else if ((card field "numNeurodes") = maxNeurodes) then
    messages "Sorry -- only" && maxNeurodes && ¬
    "letters allowed"
  else -- new letter and maximum not reached so...
    AllocateNeurode(letterNum)
    ProcessNeurode(letterNum)
  end if
  put NumToChar(letterNum+64) into currLetter
end if
```

Processing Neurodes in "Letter Learner"

Now that we know how "Letter Learner" works, we can examine a scripting technique in "Letter Learner" that does the work of allocating neurodes and adjusting the neurodes' weight vectors. The bulk of the work here is done by the script that goes with the single button that covers the entire list of letters (see Figure 11).

After doing some math that determines which letter the cursor is on top of when the user presses the mouse button, the partial script in Listing 1 is executed `on mouseUp`.

In English, this says that if the letter in question has not already been learned, allocate a neurode for that letter and then process the neurode (i.e., do all the necessary math for that neurode); otherwise, process the neurode that has already been allocated. In addition, the number of letters already taught is kept in a field called "numNeurodes." This is compared with the maximum number of letters that the network is allowed to learn at once (in this version, that number is 13). Notice that the letter being taught is stored as a number, not a character; thus, the letter E is stored in the container "LetterNum" as the value "5."

How can we tell if a letter has already been taught? Simple: If the letter has not yet been taught, the line that corresponds to that letter in the field "Weights" will be empty; otherwise it will have numbers in it. So the function `allocated`, which determines whether a particular letter has been learned by the network, reads as shown in Listing 2.

When we allocate a new neurode, all we have to do is initialize that neurode's weight vector to all zeros. The `allocateNeurode` handler does just that (see Listing 3).

`ZeroList` is a global variable that contains a string of 25 zeros, separated by commas. The field "LetterList" is the list of the letters that have already been learned that appears on the far left of the screen.

All of the necessary math is done by the `processNeurode` handler shown in Listing 4. The `DeltaVector` and `AdjustWeights` XFCN's (see box within Listing 4) deal with vector arithmetic. The `DeltaVector` handler takes the input vector and shrinks it; `AdjustWeights` takes the shrunken input vector and adds it to the weight vector for the neurode in question. This process is repeated twice, in order to give the weight vector a bit more strength.

The XFCN's `DeltaVector` and `AdjustWeights` do simple vector mathematics. These were originally implemented in HyperTalk, but HyperTalk took too long to do the math, so they were converted to *Turbo Pascal* XFCN's instead.

DrawAndGetInputVector XCMD

The biggest difficulty that I had in writing "Letter Learner" was in allowing the user to draw a letter on the screen. The first method I tried involved creating a rectangular button on the screen for the letter to be drawn in, and simply attaching the following script to the button:

```
on mouseDown
  choose paint tool
end mouseDown
```

There are actually several problems with this solution, but the biggest one is that once the paint tool is chosen, the letter drawn can extend out of the button that received the `mouseDown` message (see Figure 12). HyperTalk provides no way to restrict the users drawing within particular screen limits.

I solved the problem by writing an XCMD which I titled `DrawAndGetInputVector` (if you're not a programmer, you may not follow the rest of this discussion, but don't worry—understanding this is not necessary to be able to use "Letter Learner"). This XCMD takes the top-left and bottom-right corners of the letter button as parameters, and declares a rectangle at the same location as the button. Then the Macintosh Toolbox QuickDraw procedures are called to allow the user to draw only on the screen when the mouse is both down and within the rectangle. As the user draws the line, the information needed to derive the input vector is collected, and it is passed back to the "inputVector" field in the stack.

The five other XCMD's and XFCN's in "Letter Learner" are used to do the mathematics behind neural networks quickly and easily. Because the concepts involved in understanding neural networks are so complex, and because most people do not have intuitive knowledge of vector arithmetic, these more technical aspects of neural networks are beyond the scope of this article. *HyperCard* is a good environment for neural networks because it allows the concepts to be explained and demonstrated in a more intuitive and non-linear fashion. If your curiosity was piqued by this article, I hope you will obtain a copy of "Letter Learner" and send me any comments you have about it.

Listing 2

```
function allocated letterNum
  if line letterNum of field "weights" = empty then
    return false
  else
    return true
  end if
end allocated
```

Listing 3

```
on allocateNeurode letterNum
  global zeroList
  put "Allocating Neurode..." into field messages
  put NumToChar(letterNum+64) into letter
  put zeroList into line letterNum of field "weights"
  put (card field "numNeurodes") + 1 into card field ¬
  "numNeurodes"
  put letter into line (card field "numNeurodes") of ¬
  field "letterList"
end allocateNeurode
```

Listing 4

```
on processNeurode letterNum
  global onIcon, taughtFlag
  put "Adjusting Neurode Weights..." into field messages
  repeat with counter = 1 to 2
    put DeltaVector(field "inputVector") into field ¬
    "deltaVector"
    put AdjustWeights(field "deltaVector",¬
    line letterNum of field "Weights")¬
    into line letterNum of field "weights"
  end repeat
  put line letterNum of field "weights" into field¬
  "DisplayWeightVector"
  clearMessages
  put true into taughtFlag
  if the icon of background button "autoClear" is onIcon
  then ClearLetter
end processNeurode
```

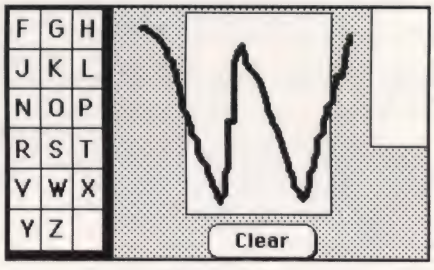


Figure 12
Here's what would happen if we used the "choose paint tool" message to let the user draw their letter in the box.

What Is a Musical Palindrome?
What Medieval Scourge Killed Petrarch's Sweetheart, Laura?
Find Out in …

Culture 1.0

Product Name: *Culture 1.0*
Company Info:
Cultural Resources, Inc.
7 Little Falls Way
Scotch Plains, NJ 07076
(201) 232-4333
Price: $175
Developers: Walter Reinhold, Chris Chapman, and Tom Tafuto

Walter Reinhold is a teacher—an extraordinary *teacher.* Now, his classroom can meet worldwide, yet each student can individually benefit from his careful tutelage and scholarship by opening a folder on a Macintosh desktop. This folder is named *Culture 1.0,* the first *HyperCard* software efforts of Cultural Resources, Inc., (CRI) of Scotch Plains, New Jersey. And, what an effort it is!

A lecturer in the *Bible* and theology, as well as many other subjects, Professor Reinhold is also an organ recitalist with numerous performances to his credit. His educational expertise is primarily in music history and style analysis, giving him a remarkably rich historical platform from which he has designed sixteen courses covering medieval to 20th century music, art, and history. He has taught thousands of students on the undergraduate, graduate, and professional levels since 1968 at New York University, and currently also teaches at the School of Music at Kean College of New Jersey.

In *Culture 1.0,* Professor Reinhold conveys his philosophical style of teaching about knowledge and perspective. He believes that one's perspective is the basis of wisdom. If one is truly to be called educated, knowing how cultural and historical facts are woven throughout the tapestry of civilization is integral to that education. To this end he has dedicated his exceptional talent to the world of scholarship. He does not fail us.

How to Begin?

First be certain your system meets the essentials. *Culture 1.0* requires *HyperCard 1.2* (or later) and a hard disk with 5 Megabytes (MB) of free space. The user's manual is simple. It carefully coaches even a novice in loading the nearly 5 MB of information (seven disks). Due to the intelligent graphic screen design and easy-to-follow manual, one can start gaining perspective by browsing history within minutes of creating the folder.

Install *Culture 1.0* following the directions precisely. When all 10 stacks are displayed, begin by double clicking on the "Overview" stack. From the overview screen (see Figure 1), one can navigate all the components of *Culture 1.0.* Reading the overview notes and then displaying the roster of essays provides the browser with the essence of the program.

People, events, dates and facts—the chrysalis of western civilization—come alive, and Walter paints a

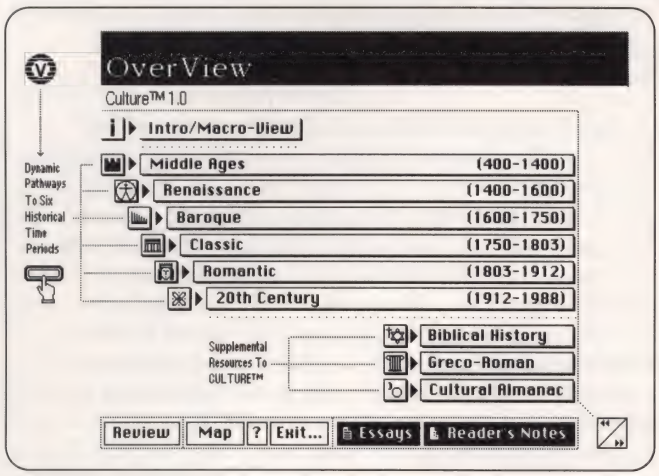


Figure 1
This "Overview Card" is the takeoff point for Culture 1.0. *It is virtually always a single mouse click away, so the user can return to it to launch a new exploration of a different time, place, or subject matter.*

contextual canvas with a masterful touch of organization. The specific eras and generations form the chronological timeline, and these are further delineated by country and discipline. The framework of *Culture* helps you to experience the most important composers, painters, architects, authors, leaders, and rulers in each era. You also see many great works of art and architecture (there are over two hundred graphic images in *Culture).* Plus, you can listen to 75 signature melodies of the composers. One drawback to the music, however, is the sound quality. Due to space limitations (yes even 5MB is not enough), the sounds were sampled at a relatively low sample rate (11KHz), and the quality is not as pure as it could be. For you audiophiles out there, the samples contain some aliasing.

Professor Reinhold has retained an enormously enthusiastic and gifted talent for teaching. One that translates well to the Macintosh. This talent reaches right out through the Mac screen and keeps one clearly focused on learning. His ebullient personality truly radiates through the introductory essays, the 30 general essays that introduce each historical period with specific references to music and art; plus, 59 period essays on people, events, and interrelated aspects of history. He writes concisely, with a wit and energy that we may all be uniquely touched by. The time periods of study are the Middle Ages, Renaissance, Baroque, Classic, Romantic, and 20th Century. Each era also has a "popmenu" listing the sub-periods available for study. Additionally, three supplemental resources—Biblical History, Greco-Roman, and a Cultural Almanac can also be accessed for even more in-depth information.

Perhaps 20 years of devoted scholarship and a lively, intelligent and humorous viewpoint keep this compendium of historical facts, grids, and cultural timelines vibrant among the many cultural disciplines Professor Reinhold enthusiastically shares with us. We, as eager "magnetic media" students, transcend time from the Ancient Greeks up until the 20th Century. It is a joy.

Customization Features

Culture 1.0 incorporates three very important features that let you make its stacks your own personal reference:

- First, its "Culture Links" button allows you to create/delete customized links to other relevant cards of your own choosing.
- Second, its "Smart Paths" structuring capability allows custom navigational path or custom lists of an associative network of cards (bridging all 10 stacks in *Culture*) to be generated. If one were researching a specific topic over several disciplines, the "Smart Path" option would be invaluable. *Culture's* programmers have made it easy for you to follow the four steps required to load, save or clear a specific path.
- A third major feature is the "Reader's Notes" button allowing individual thoughts or comment additions to any card at any time. With the normal *HyperCard* 30,000 character per field limitation, that should be sufficient for anyone's needs. User/reader notes can be saved as a text file for importing into a word processor for printing. This "Do Reports" function is versatile enough to save notes for a single card, or all of your notes for an entire time period.

"Knowledge Navigation" Pioneers

With *HyperCard's* inherent linking facility, CRI's programmers have structured over 2,000 word links allowing generous free-play of the mind. Talented programmer Thomas Tafuto of Applied Imagination showed some incredible skill in software design building this product. Employing the HyPict XCMD, written by Martine Heinsdorf, much of the art can be proportionally and automatically re-sized into an expandable, floating window—the larger the window, the larger the art. (see Figures 3, 4, and 5)

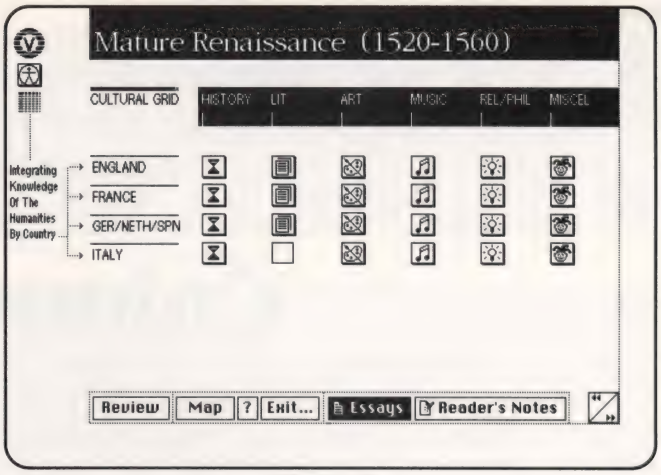


Figure 2

The Cultural Grid makes it easy for the Culture 1.0 *user to explore a period of history both by country and by discipline. Comparisons and contrasts are easy to make between both diverse and closely related times and places.*

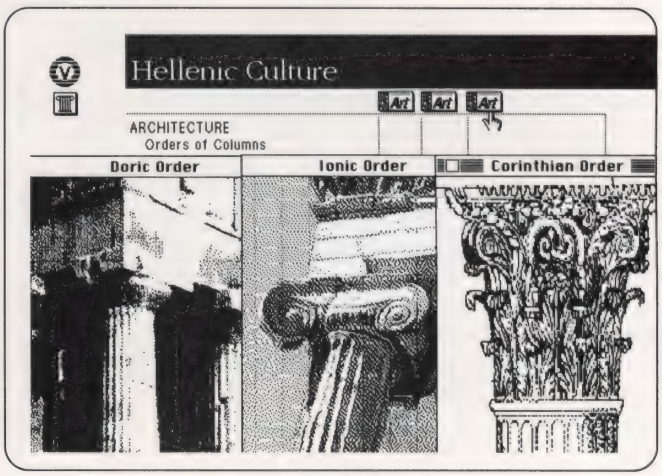


Figure 3

Here three HyPict windows are displayed allowing the user to compare three Hellenic Orders of Columns.

Another nifty feature is *Culture's* quick scan capability. Need to quickly scan all cards in a specific stack? Just use the left or right arrow button at the corner of each card and you can navigate easily. Exit by clicking the mouse button.

Recommending this here-and-now educational workstation for gaining perspective on what has shaped western culture is a pleasure. This is a college extension course for a very minimal price. Curiosity is the only prerequisite. Every student and educator should have access to this involving database of the humanities.

Thanks to Walter Reinhold, Chris Chapman (the student that discovered him and combined his knowledge and HyperCard), and the devoted team at CRI and Applied Imagination for their considerable effort to bring this product to market at a time when software of this calibre is essentially nonexistent. They win all awards I could possible give. Return to this program when negative stress starts to encroach on your personal creativity. An hour spent with *Culture 1.0* will refresh and inspire you to put those nagging business concerns

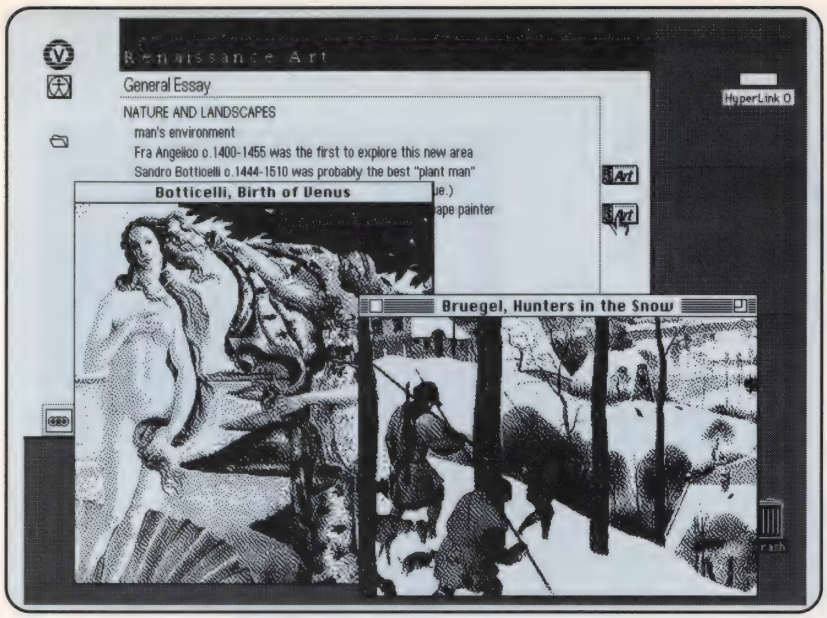


Figure 4

The HyPict XCMD opens up the possibility of a much larger screen display to HyperCard. *The screen shot here is scaled to exactly the same percentage as the others in this review, but because it came from a Macintosh II, it is much larger. Notice that two different paintngs are displayed, each in its own expandable floating window. The artwork is limited to the normal 72 dots per inch (dpi) of all* HyperCard *artwork. CRI plans to make use of Apple's* HyperTV *as it becomes available, which will allow for better resolution, color, and even laser disc quality.*

aside so that you may get on with the creative pursuits in your life. Yes, *Culture* will relax and inspire you to know more, to appreciate more, and serve as a healthy reminder to never give up—whatever mundane state you may be in at present—that aggressive search for learning. Knowing that your search will soon bring you to a more appreciative viewpoint of life.

When you shut down the Mac for the night, I'm quite sure it purrs more contentedly with *Culture 1.0* in a folder happily nested with your more statistically oriented software. Yes, culture has come to the desktop. It is about time. Walter, your worldwide classroom awaits!

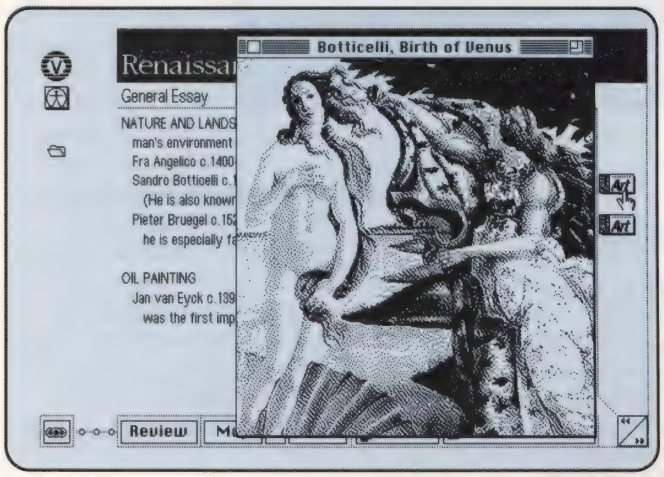


Figure 5

Here the same Botticelli as shown in Figure 4 is displayed on the smaller Macintosh screen. The HyPict XCMD works equally well on all Macintosh screens.

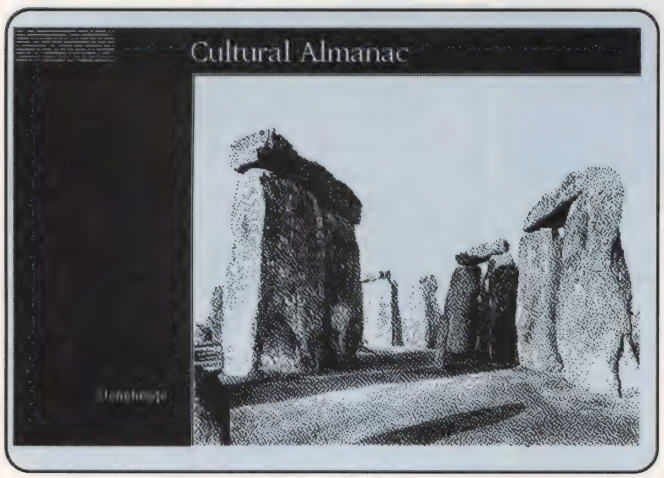


Figure 6

Each section of Culture 1.0 *is introduced with scans of significant historical interest.*

Culture 2.0 Anyone?

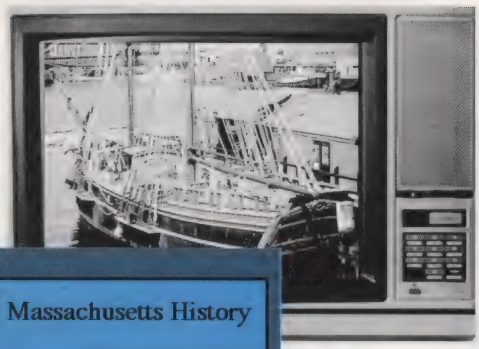
The Mac community is indeed fortunate to be among the first to benefit from CRI's initial software developments. With this product as their hub, they plan expansion to a *Culture 2.0* version on CD-ROM that will give listeners improved sound quality. The Interacting Muse—a series of six CD-ROM disks that feature the artists, sculptors, architects, musicians, and other historical figures from the six time periods talked about in this overview. The Music Tours—a *HyperCard* blueprint of musical architecture from these same six eras. And, last but not least, a planned newsletter for cultural information sharing—hoping to attract authors of complementary stacks and scripts.

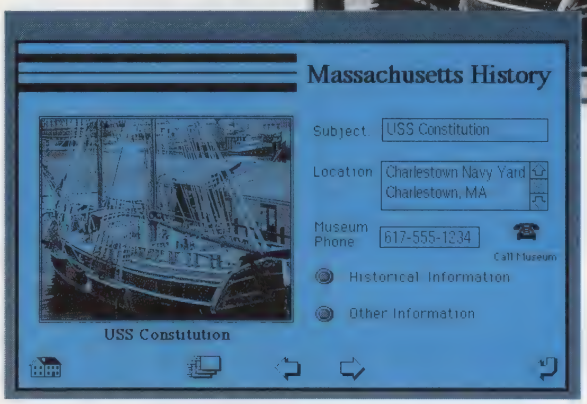
As we go to press, CRI announced future developments for *Culture*. These will include additional structural levels (new types of knowledge, non-Western cultures, greater level of detail within the cultural database; a powerful authoring system for those who do not understand HyperTalk; and, color graphics and orchestral sounds. According to company chairman, Chris Chapman, "The company is pursuing the utilization of *HyperTV* videodisk technology in order to present Walter Reinhold as an animated interactive guide to the program." [For those of you who haven't heard of this future development previewed by Apple's John Sculley at MacWorld Expo in San Francisco, *HyperTV* incorporates the AST video digitizer board to produce real-time video images from video tape, laser disc, or live, within a window right on the *HyperCard* screen.—Ed.]

—Carole Eversole

INTRODUCING HYPERVISION™

for the ProViz™ Video Digitizer. The first Real–Time video software designed exclusively for HyperCard.™





Real–world images increase the educational power of HyperCard.

Video to HyperCard...

Now you can capture virtually any video image *directly* into your HyperCard stack, without the need for still models or a freeze-frame. With just one click of the mouse you'll instantly grab images from camcorders, VCRs, laser disk players—even television.

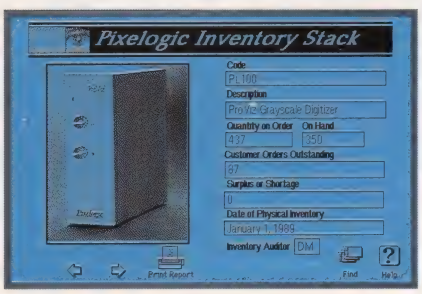
HyperVision is the new software interface to HyperCard for the line of ProViz Video Digitizers. Because HyperVision was developed specifically for the ProViz Digitizer and incorporates Apple's HyperScan™ software, you can capture and display images with unparalleled clarity.

Real-time and more...

- Real-time image capture for frame-grabbing or live pictures.
- Image scaling and cropping.
- The dithering algorithms from Apple's HyperScan software.
- The extra features of the ProViz software, including image capture in 16 shades of gray and 256 shades of gray.

Real–time images bring personnel stacks to life.

Real images bring product information into focus.



Just imagine what you can do with real-world images in your stacks:

- Add frames to the HyperCard front-end of an interactive videodisc.
- Supply your customers with an electronic sales catalog.
- Add product shots to your on-line documentation and demo disks.
- Incorporate photos into your electronic mail, personnel, or security system.

...on the entire Macintosh family.

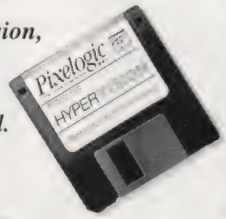
All you need to run HyperVision and the ProViz is:

- Any Macintosh—a Plus, SE, SE/30, II, IIx, or IIcx—with 2 Mb of RAM and a hard disk.
- System version 6.0 or later
- HyperCard version 1.2 or later.

Free Demo Disk

Our self-running demo disk will show you the simplicity and power of **HyperVision**. Call our toll-free number today and see for yourself why we say:

With HyperVision, HyperCard never looked so good.



1•800•432•2292

(in Massachusetts 617•938•7711)

Pixelogic℠

The Vision of the Future℠

Scanning Is a Snap with Your Video Camera,

Proviz & HyperScan

By David G. Brader

If you need more scanning flexibility than a flat-bed scanner provides, check out Pixelogic's ProViz line of video digitizers. They offer both grayscale and color video digitizer systems. The ProViz Video Digitizers can scan anything a flatbed scanner can and more. We are just starting to use our units, and I want to share some of our excitement with you—including our neat "Capture" button that captures images for *any* stack using Pixelogic's *HyperVision* stackware.

When the ProViz hardware arrived it certainly was tempting to start playing with the color digitizer immediately (with the Mac II), but I have held off until we have the time to thoroughly check out the system with Silicon Beach's new *SuperCard*. Besides, the *ProViz HyperVision* stack looked interesting.

HyperVision is a modified and enhanced version of Bill Atkinson's *HyperScan* (licensed from Apple Computer by Pixelogic) [See David Brader's article "*HyperCard* and the Apple Scanner" in the Nov/Dec 1988 issue of *HyperLink* for details on using *HyperScan* with the Apple Scanner—Ed.] The Apple Scanner driver software has been removed and the ProViz Digitizer driver software added. Pixelogic also modified the look and feel to be more appropriate to the ProViz Video Digitizers.

All the high-tech toys—video cameras, VCR's, video disk plays, and TV's—can provide images for *HyperCard* stacks when you use a ProViz Digitizer and *HyperVision*. Imagine watching a video on your VCR and, at the exact moment you see a scene appear on the TV monitor, pressing a *HyperCard* "Snap" button that captures that frame of video. A couple of mouse clicks and minutes later that image appears in your *HyperCard* stack, and you are ready for the next take.

What You Need to Make It a Snap

You should have at least a Mac Plus with a hard drive and *HyperCard* 1.2 or later to work with the ProViz Digitizer and *HyperVision*. Of course, a video source is required, such as a VCR with an NTSC video output. My setup for this article was a Mac Plus with 4 megabytes of RAM, a hard drive, the Proviz Video Digitizer (see the self portrait above), a TV monitor with a video input jack, and an 8mm Sony video camera/VCR. These hand-held units are great for real estate brokers and others that require video capture for *HyperCard* stacks.

A Self Portrait of
The *ProViz Grayscale Video Digitizer*

A Universal Picture-Capture Button

The "Capture" button I'm presenting here does two things. First, it defines, by its size and position, where the graphic will be placed on a card. Second, once activated it clears its space on the card, opens the *HyperVision* stack, and passes its position and dimensions to *HyperVision*. This button becomes the target for the save image function from *HyperVision*. This means that any image captured ends up over the "Capture" button on the original card.

The "Capture" button can be cloned, moved to any stack, pasted in a background or card level, and resized freely. If multiple "Capture" buttons are placed on one background or card, only the single button activated receives the video image. Let your imagination run free and see what video capture applications you dream up.

To allow the "Capture" button to function, *HyperVision* must be modified. The modifications do not change how *HyperVision* works by itself. This is

David G. Brader, HyperLink Magazine's *Publisher, has worked with computers for over two decades.*

because, when *HyperVision* receives an `openStack` message, if my special global variable, `where`, is `empty`, *HyperVision* functions just like the Pixelogic manual says it should.

Button Down Logic

When you activate the "Capture" button from your stack, it opens the *HyperVision* stack. The `openStack` script in the modified *HyperVision* stack detects that it was entered via a "Capture" button (because the global variable `where` contains `here` instead of being `empty`). This modified stack script prepares the "Copy Stand" card (see Figure 2) or the "Video Scan" card (see Figure 7) to show a special "Save Area" button over each of the cards' normal "Scan Area" buttons. The normal "Scan Area" button is always proportional to a full *HyperCard* window. The newly visible "Save Area" buttons are proportional to the "Capture" button that was activated. The "Save Area" button is scaled the same way as its companion "Scan Area" button.

A "Save Area" button is always located in the upper-left corner of its companion "Scan Area" button. You resize (or position) both buttons simultaneously by dragging the corners (or middle) of the larger "Scan Area" button. (An operational note: When in the "Capture" button mode, do not use the "Scale Setting" field to adjust the scale because the scale relationship between the "Save Area" and "Scan Area" buttons will be lost.)

The portion of the previewed image that appears within the smaller "Save Area" button rectangle is the portion of the image that will be moved back to the stack containing the active "Capture" button.

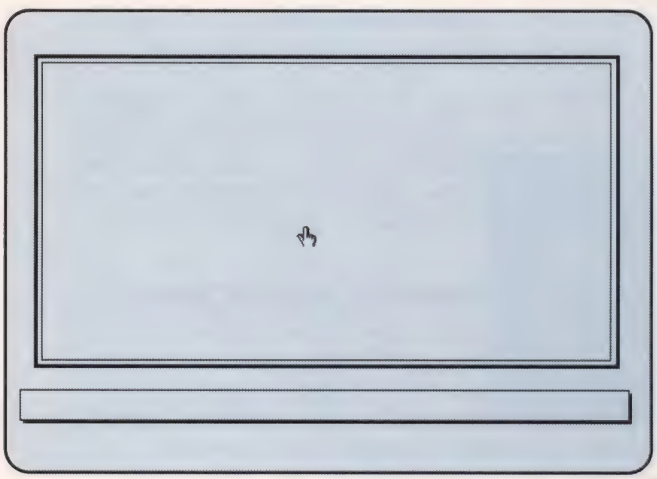


Figure 1
A "Capture" button fills the frame of this Photo Album card.

Figure 1 shows a typical card with a "Capture" button sized and positioned to fill the inside of the picture frame graphic. In this "Photo Album" stack example, the graphic picture frame, "Capture" button, and picture title field are all placed in the background. By using the "New Card" command from the "Edit" menu blank photo album cards can be added. Clicking on the area inside of the picture frame on one of these cards activates the transparent "Capture" button.

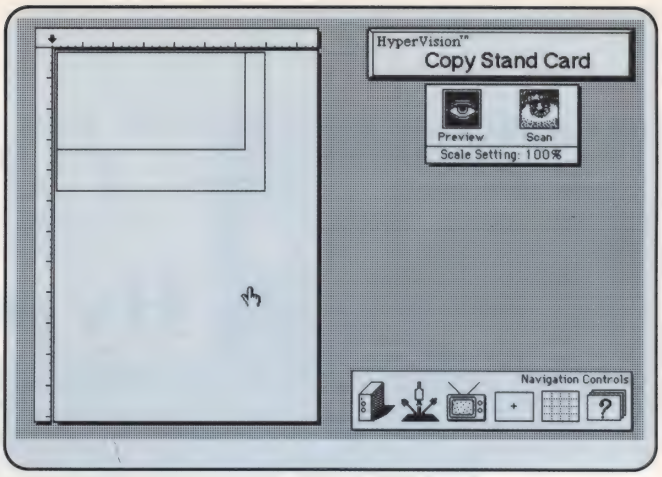


Figure 2
The "Scan Area" is covered by a smaller "Save Area" button in the upper-left corner.

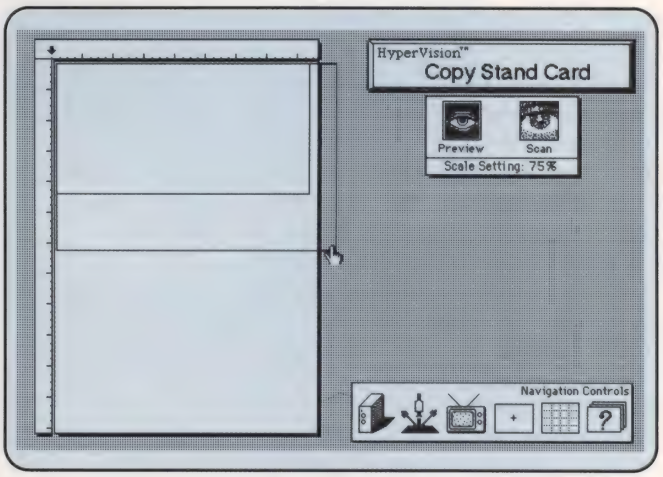


Figure 3
After resizing the two rectangular buttons on the image area.

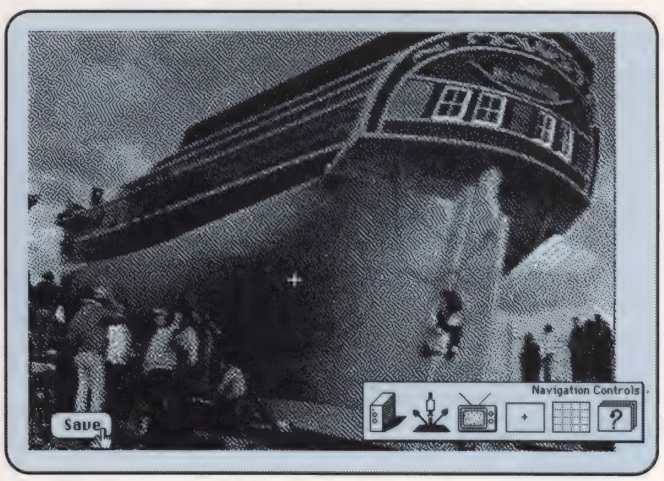


Figure 4
This image, caputured on the "Halftone" card, is ready to save.

Figure 2 shows the *HyperVision* stack "Copy Stand" card with its "Scan Area" button rectangle overlaid with the smaller "Save Area" button rectangle. Figure 3 shows the same card after resizing and positioning the buttons to capture the portion of the image desired. The "Preview" button is activated as many times as

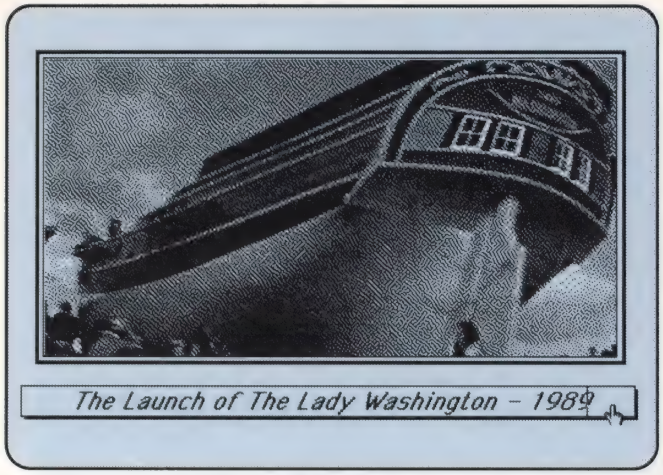


Figure 5
The image has been pasted over the top of the "Capture" button.

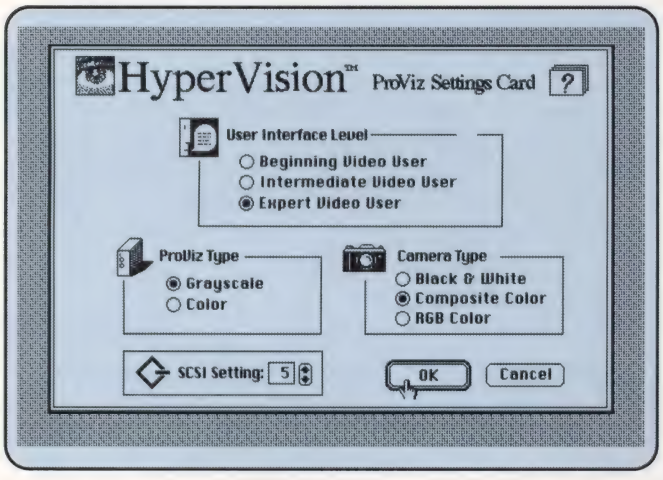


Figure 6
This card allows you to configure your video capture system.

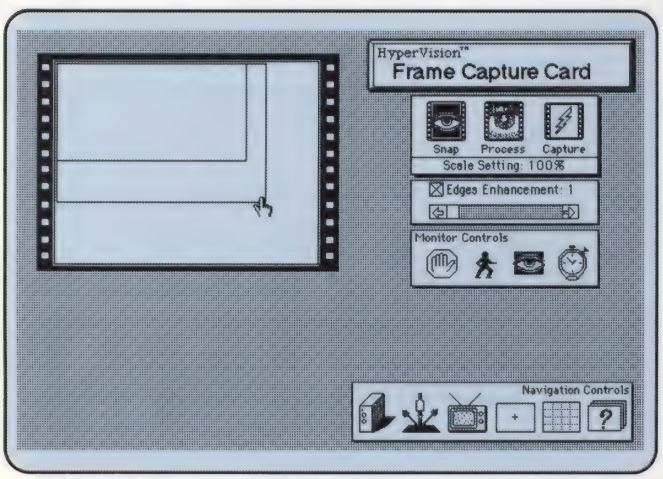


Figure 7
The "Video Scan" card is used for live video frame capture.

required to adjust the ProViz settings and video camera position. When you are satisfied with the settings, activate the "Scan" button to capture a frame of video. After about a minute of processing, the "Halftone" card appears with the captured image ready for any adjustments or enhancements as explained in the *HyperVision* manual (see Figure 4). Finally, press "Save" on the "Halftone" card, and the *HyperVision* stack compacts itself, copying the proper image area. You are then returned to the stack and card containing the activated "Capture" button where the image is pasted and positioned over the "Capture" button on the card level (see Figure 5). Now you could go to a different album page and paste another image or activate the "Capture" button from the same card and replace the image just pasted.

Creating the "Capture" Button

The description of the "Capture" button is listed below. Remember, once you have built one of these "Capture" buttons, it can be copied and pasted in any stack at either the card level or the background level. After you have positioned and sized it with the Button tool, choose the Browse tool and the new copy of the button is "locked, loaded, and ready to rock and roll."

Background Button: or **Card Button:**
Capture
From Background:----------
or **From Card:**----------
AutoHilite: false
ShowName: false
Visible: true
Icon: none
Rectangle: position & size for your picture
Style: transparent
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: Yes - See Listing 1

Listing 1

```
on mouseUp
 global where,xStart,yStart,xSize,ySize
  answer "Do you want to scan" && ¬
  "a new image?" with "Yes" or "No"
  if it is "Yes" then
    get the rect of me
    put item 1 of it into xStart
    put item 2 of it into yStart
    put item 3 of it into xEnd
    put item 4 of it into yEnd
    put xEnd - xStart into xSize
    put yEnd - yStart into ySize
    choose pencil tool
    drag from xStart + 1, yStart + 1 to ¬
    xStart + 2, YStart + 2
    choose select tool
    drag from xStart,yStart to ¬
    xEnd,yEnd with optionKey
    doMenu "Cut Picture"
    choose browse tool
    push card
    put "here" into where
    go to card "Copy Stand" of stack ¬
    "HyperVision"
  else
    put empty into where
  end if
end mouseUp
```

HyperVision Modifications

First, open the *HyperVision* stack and go to the "Copy Stand" card. Select the Button tool from the Tools menu. Double-click on the rectangle outlined in the white scan area. Make sure this is the "Scan Area" button by checking the name in the "Button Info..." dialog box. Close the box and, while the button is still selected, choose the "Copy Button" command from the Edit menu. Now, Choose the "Paste Button" command from the Edit menu. Double-click on the button again to bring up the "Button Info..." dialog box of the copied button. Change the name to "Save Area" and replace the script in its script window with the following:

```
on mouseDown
  send mouseDown to bg btn "Scan Area"
end mouseDown
```

Repeat this whole procedure to the scan area on the "Video Scan" card in the *HyperVision* stack.

Listing 2 shows the additions to the *HyperVision* stack script; the changes are in the boxed area. Replace the script in the "Scan Area" button on the "Copy Stand" and "Video Scan" cards with the script in Listing 3 ,then go to the "Halftone" card and replace the existing "Save" button's script with the script in Listing 4.

Listing 2 - Modifications for the stack script

```
on openStack
  global brightness,contrast,oldBrightness
  global oldContrast,walkThrough,¬
  enhanceEdges
  global where,xStart,yStart,xSize,ySize
  if where is not empty then
    put xSize div 3 into width
    put ySize div 3 into height
    show bkgnd button "Save Area" ¬
    of card "Copy Stand"
    get rect of bkgnd button "Scan Area" ¬
    of card "Copy Stand"
    put 32 into item 1 of it
    put 27 into item 2 of it
    put 32 + width into item 3 of it
    put 27 + height into item 4 of it
    set rect of bkgnd button "Save Area" ¬
    of card "Copy Stand" to it
    set rect of bkgnd button "Save Area" ¬
    of card "Video Scan" to it
    put "32,27,203,141" into it
    set rect of bkgnd button "Scan Area" ¬
    of card "Copy Stand" to it
    set rect of bkgnd button "Scan Area" ¬
    of card "Video Scan" to it
    scale 100
  end if
  set cursor to watch
  --Leave the rest of this script as is
end openStack

on closeStack
  global Install,settings,ProViztype,¬
  Camera,AddressLoc,where
  if Install is "on" then exit closestack
  put empty into where
  hide bg btn "Save Area" of card ¬
  "Copy Stand"
  hide bg btn "Save Area" of card ¬
  "Video Scan"
  show menuBar
  get ProViz(KillBuffer)
  get HyperScan("CloseScanner")
  writesettings
  compactstack
end closeStack
```

Listing 3 - Script for the "Scan Area" buttons

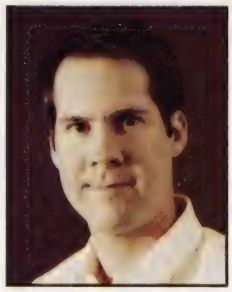
```
on mouseDown
  global where, xSize, ySize
  hide me
  if where is not empty then
    hide bg btn "Save Area"
    get the rect of me
    put item 3 of it-item 1 of it into oldX
    put item 4 of it-item 2 of it into oldY
    put "32,27,201,268" into pinRect
    get HyperScan("DragRect",¬
    rect of me,pinRect)
    set the rect of me to it
    put item 1 of it into item 1 of newRect
    put item 2 of it into item 2 of newRect
    put xSize * (item 3 of it - ¬
    item 1 of it) div 512 into newX
    put newX + item 1 of newRect into ¬
    item 3 of newRect
    put ySize * (item 4 of it - ¬
    item 2 of it) div 342 into newy
    put newY + item 2 of newRect into ¬
    item 4 of newRect
    set rect of bkgnd Button "Save Area" ¬
    to newRect
    show bg btn "Save Area"
  else
    put "32,27,201,161" into pinRect
    get HyperScan("DragRect",¬
    rect of me,pinRect)
    set the rect of me to it
  end if
  show me
  scalegauge
end mouseDown
```

Listing 4 - Script for the "Save" button

```
on mouseDown
end mouseDown

on mouseUp
  global where,xStart,yStart,xSize,ySize
  if where is not empty then
    put empty into where
    domenu compact stack
    choose select tool
    drag from 0,0 to xSize,YSize ¬
    with optionKey
    doMenu "Copy Picture"
    pop card
    choose select tool
    doMenu "Paste Picture"
    drag from 0,0 to xStart,yStart
    choose browse tool
    hide menuBar
  else
    set hilite of me to true
    set cursor to busy
    saveCard
    set hilite of me to false
  end if
end mouseUp
```

Xpanding HyperCard



A Look at Two Commercial Products That Expand HyperCard's *Horizons*

by James Paul

In the last two "Xpanding HyperCard" columns, we explored the *HyperCard* Toolbox and how it works. Now that we are better acquainted with the way XCMD's and XFCN's work, let's take a look at what's going on in the commercial marketplace. We'll look at how *VersaCad* is using *HyperCard* to enhance Computer Aided Design, and how Symmetry has given a piece of *HyperCard* to regular applications.

HyperCad

VersaCad Corporation has added an interesting and powerful ability to *VersaCad/Macintosh Edition*. Using XFCN's, *VersaCad* provides the ability to read drawing information into a *HyperCard* stack, process and modify it, and save it back to disk. *VersaCad* comes with stacks that can use drawing information to calculate, for example, bills of materials (see Figure 1), or door and window schedules. In fact, there is a "Parametric Design" stack that can allow *VersaCad* drawings to be displayed and altered from within *HyperCard,* with changes saved back to the *VersaCad* file. Through the use of the *HyperCad* XFCN's, *HyperCard* becomes a powerful extension to *VersaCad*.

> *Through the use of the* HyperCad *XFCN's,* HyperCard *becomes a powerful extension to* VersaCad.

Out of the box, the *HyperCad* stacks are significantly powerful. Also, they can be "tweaked" for specific projects fairly easily, since they are regular stacks. What makes *HyperCad* shine, however, is that new stacks can be written and completely customized for nearly any type of CAD work. Details about how the XFCN's work are provided, along with examples. VersaCad Corporation is strongly encouraging us to make our own stacks, putting the power and flexibility of *HyperCard* and *VersaCad* together for individual users. Also, they are sponsoring a *HyperCard* Design Stack contest to further inspire people to find new ways to use `HyperCad`.

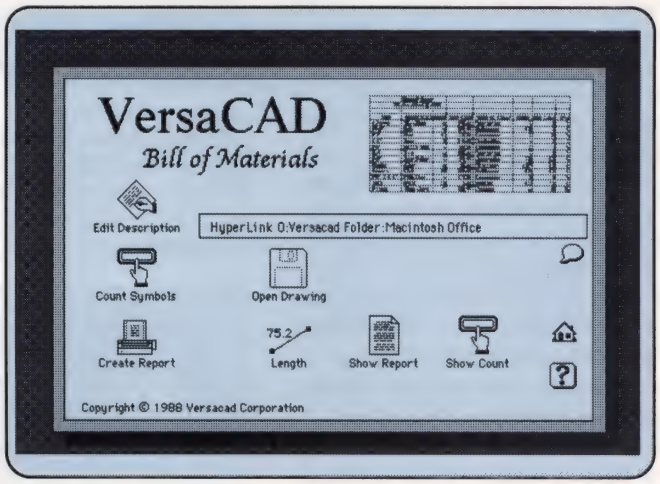


Figure 1
The first card of VersaCad's *"Bill of Materials" stack that allows manipulation and cataloging of the elements of a* VersaCad *drawing from directly within a* HyperCard *stack.*

How *HyperCad* Works

At the heart of *HyperCad* are two XFCN's. Yes, that's right, only two. One XFCN is named `getvcad`, and the other is called `putvcad`. It's not hard to figure out what these two commands do. The `getvcad` function reads information from a *VersaCad* drawing file, and the `putvcad` function saves drawing information back to a *VersaCad* drawing file. These XFCN's are very general, leaving maximum flexibility in the hands of the person writing the stack.

VersaCad drawing files are made up of records. There are several different types of records, each type has its own format and specific information. There are records types that hold drawing information, saved windows, plot specs, object data, symbol control info, and symbol object data. If these don't mean much to you, don't worry, they are explained in detail in the *VersaCad* manual. Also, it's possible that only one or two of

James Paul is chief programmer for Paul Software Engineering, and he is the author of Icon Factory *from HyperPress. He is also the author of numerous XCMD's and XFCN's, which include DoList, ListRes, and SortIt.*

the record types is relevant for a specific project, making it possible to concentrate on fewer things at a time while writing a stack.

Using the `getvcad` XFCN, we can get a specific record from a drawing file, and put it into a *HyperCard* container. Each line in the container has one piece of information from the record. The line numbers for each piece of data are included in the information provided by VersaCad Corporation. After we have read a record, we can pick and choose the pieces of data we want to work with. We can also change any part of the data and save it back to the file using the `putvcad` XFCN.

The *HyperCard* stacks that come with *VersaCad* are perfect examples of what can be done with the XFCN's. For example, there is a "Bills of Materials" stack (see Figure 1) that lists the materials needed for production of a *VersaCad* and calculates the cost of the materials. The stack manages this by first using the `getvcad` XFCN to read information about the objects in the drawing. Then, it lets the user manipulate the information in typical *HyperCard* fashion (see Figure 2). A report feature is also available which saves all the data in a text file which can be printed from a word processor. Changes to the drawing can also be saved back to the drawing file using `putvcad`. As usual, a good way to learn about how these stacks work is to examine them carefully.

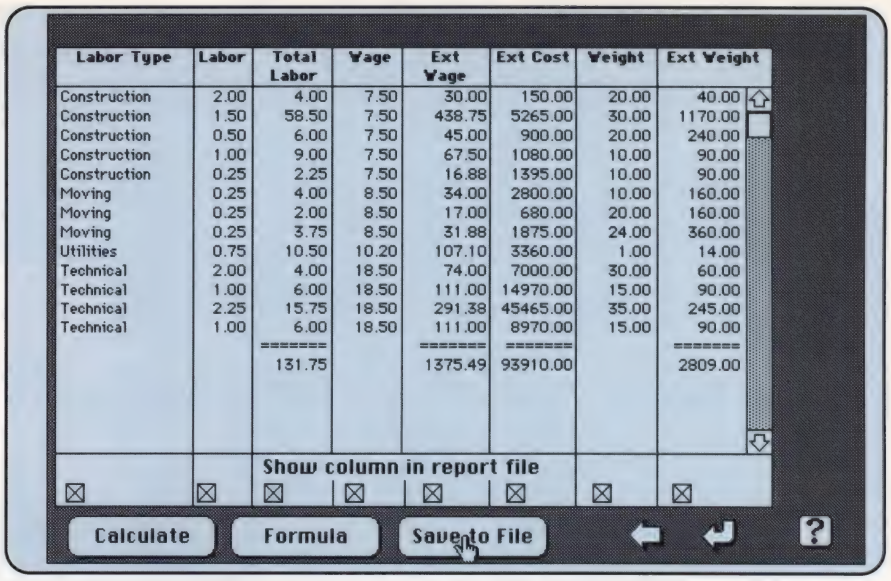
The flexibility of these XFCN's is too great for me to describe here in more detail. The *HyperCard* Design Stack contest I mentioned earlier is for people who have specific ideas for using HyperCard's ease of use for extending the productivity of *VersaCad*. To enter the contest, you should get a copy of *VersaCad/Macintosh Edition*. Developer pricing is available (at a 90% savings) for those who qualify. For more information, contact:

VersaCad Corporation
2124 Main Street
Huntington Beach, CA 92648
(714) 960-7720

HyperEngine

A product called *HyperEngine* is being published by Symmetry Corporation that gives application programmers simple access to *HyperCard* flexibility. Application programmers are people who write stand-alone programs. *HyperCard* can easily open an application from a stack, and many people have made use of this ability. Applications, however, cannot easily open *HyperCard* stacks. Until *HyperEngine* came along, that is.

HyperEngine provides a set of Toolbox routines for programmers to use when writing an application. [See this column in the Jan/Feb and Mar/April issues for information about Toolboxes—Ed.] These routines can open and close stacks from the application. The only drawback is that the application must be written to use these routines. This means that an existing application would have to be re-written to some extent to use *HyperEngine*. If you don't write applications, *HyperEngine* may not be something you are likely to find useful. Symmetry does



Labor Type	Labor	Total Labor	Wage	Ext Wage	Ext Cost	Weight	Ext Weight
Construction	2.00	4.00	7.50	30.00	150.00	20.00	40.00
Construction	1.50	58.50	7.50	438.75	5265.00	30.00	1170.00
Construction	0.50	6.00	7.50	45.00	900.00	20.00	240.00
Construction	1.00	9.00	7.50	67.50	1080.00	10.00	90.00
Construction	0.25	2.25	7.50	16.88	1395.00	10.00	90.00
Moving	0.25	4.00	8.50	34.00	2800.00	10.00	160.00
Moving	0.25	2.00	8.50	17.00	680.00	20.00	160.00
Moving	0.25	3.75	8.50	31.88	1875.00	24.00	360.00
Utilities	0.75	10.50	10.20	107.10	3360.00	1.00	14.00
Technical	2.00	4.00	18.50	74.00	7000.00	30.00	60.00
Technical	1.00	6.00	18.50	111.00	14970.00	15.00	90.00
Technical	2.25	15.75	18.50	291.38	45465.00	35.00	245.00
Technical	1.00	6.00	18.50	111.00	8970.00	15.00	90.00
		131.75		1375.49	93910.00		2809.00

Show column in report file

Calculate | Formula | Save to File

Figure 2

The "Bill of Materials" stack not only extracts information from VersaCad *drawing documents, it also sets up some excellent data storage and manipulation templates. Here, by counting objects in a* VersaCad *drawing and assigning parts and labor values to each object type a complete cost estimate of the project is generated in* HyperCard.

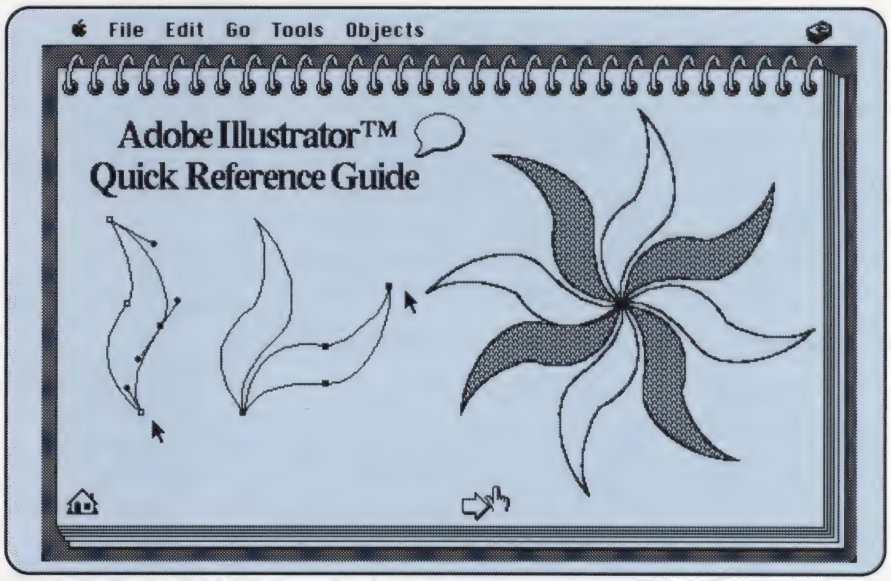


Figure 3

Adobe Illustrator is one of the many products that are turning to HyperCard *and* HyperEngine *to integrate stacks as the cornerstones of their on-line help systems.*

have a product called *HyperDA,* a Desk Accessory that can open stacks and browse through them. *HyperDA* is in effect a DA that uses *HyperEngine.*

What Can It Do?

HyperEngine can open a stack, and allow you to browse through it. There are some limitations to keep in mind, mainly that stacks can only be browsed and cannot be altered. Not all of HyperTalk is supported by *HyperEngine,* so things in some scripts may be ignored. Among things not supported are: Playing sounds, visual effects, looping, arithmetic, XCMD's and XFCN's, and any HyperTalk command that changes something in the stack. Stacks that use unsupported scripts won't bomb, but the unrecognized lines of scripting will be ignored. At first, this may look like a major drawback, and you may wonder why someone might use *HyperEngine* when so many stacks obviously depend on these unsupported HyperTalk commands. Rest assured, *HyperEngine* can be very useful if it's used to implement something every application has. Or rather, should have—on-line help.

An on-line help system is one where information about the program and its features is available from the program itself while it is running. It takes a lot of effort to write an on-line help system for an application, and many programmers just don't do it. Also, in many cases, the skills required to write a good program are very different from the skills needed to write good and easy-to-use on-line help. *HyperEngine* help solves these problem nicely. The application can be written to use *HyperEngine* for a help system (see Figure 3). Then, after the application is finished, a *HyperCard* stack can be created to serve as the help stack. The help stack could even be written by somebody who doesn't know the first thing about writing an application, but is good at writing manuals and on-line help. But what about the limitations of *HyperEngine,* you ask? Well, an on-line help system doesn't need to use much HyperTalk, and is perfectly suited to a browsing environment.

How It Works

HyperEngine is a collection of two functions and four procedures, six routines in all. It can be used easily from either Pascal, C, or assembly language. The routines perform the following functions: Open a stack, close a stack, go to a card with a specific ID, and handle an event for the stack. There is also a routine that needs to be called in the main event loop of the application as often as possible, and a function that returns error codes.

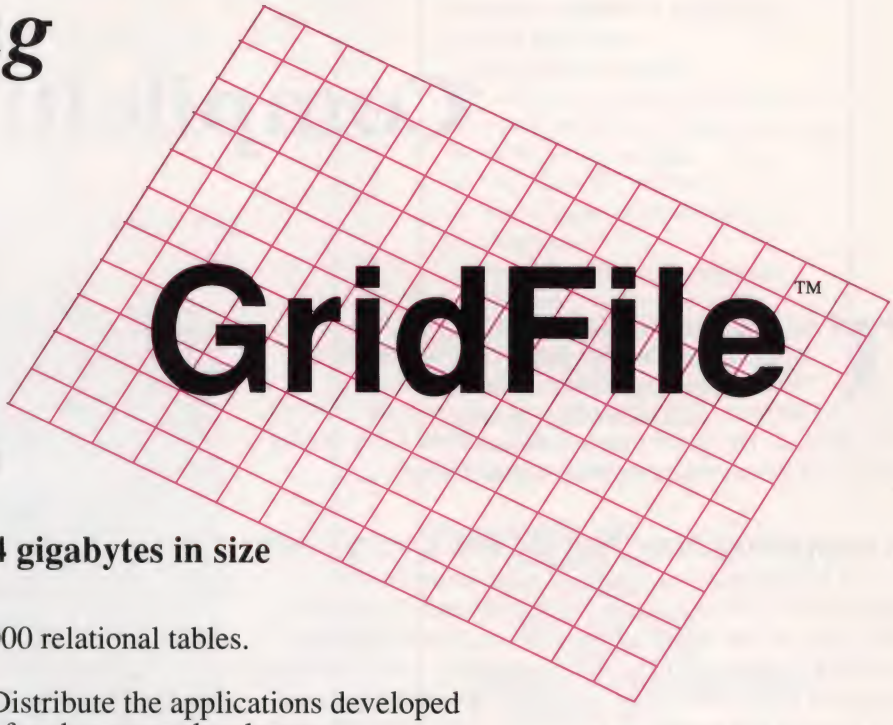
For stack writers, the *HyperEngine* manual lists all the HyperTalk commands supported by *HyperEngine*, and offers ideas and tips for writing stacks to work specifically with *HyperEngine*. These tips and ideas are also valuable if you would like your stack to be as useful as possible when it is opened with *HyperDA*. For more information about *HyperEngine*, contact:

Symmetry Corporation
761 E. University Drive
Mesa, Arizona 85203
(602) 844-2199

The technical details about *HyperEngine* and *VersaCad's HyperCad* XFCN's are beyond the scope of this column, and it's been my intention only to describe what the products are, giving examples of how they might be used. I believe that both of these products are significant in how they expand the flexibility and potential of *HyperCard*. During the last 18 months I've watched the number of *HyperCard* stacks grow tremendously, and *VersaCad* is the first truly high-powered application I have seen use *HyperCard* in such a natural and extremely productive way. *HyperEngine* approaches productivity from an interesting and useful angle. I'm aware of some major software developers who are looking at *HyperEngine* for implementing their on-line help. These two products are just two examples of how the *HyperCard* concept is working its way into all types of Macintosh productivity software.

Attention All Macintosh Developers and Users

Announcing



- **Fully relational database**
- **Multi-key indexing**
- **Compiled field constraints**
- **Supports databases up to 4 gigabytes in size**
- Each database may have up to 1000 relational tables.
- No runtime fees for developers. Distribute the applications developed without additional charges. Ideal for shareware developers.
- Fully relational database access including transparent join and select.
- GridFile databases are designed with an interactive tool which allows you to rapidly develop intricate databases.
- Performs consistency checks for intersecting fields of joined relations during all update and insertion operations.
- Supports field types of byte, integer, long integer, floating point, character string, and numeric string. Strings are variable in size and may be up to 1024 bytes for character strings and up to 40 bytes for numeric strings.
- Binary data of any size may be attached to each record. Limited only by the size of memory, this feature can be used to store PICT's, bitmaps, and other Macintosh items.
- Up to 10 times faster than B-Tree databases. Fully specified records are found within two disk reads regardless of the database size.
- Available for **HyperCard**,™ **SuperCard**,™ and **LightSpeed C.**™ Other languages available soon.

For additional information contact:

NovaSoft Engineering Group, Inc.

2343 S. Ridgewood Ave. • Edgewater, FL 32032 • (904) 423-5189

FAX (904) 428-0765

How Does the First HyperTalk Compiler Meet the Needs of Scripters?

CompileIt!

The opening screen of *CompileIt!* says it is "The Script Compiler," but it might better be called "The *Handler* Compiler." The limitations in the present version (1.1) might be enough to scare off the uninitiated. The good news is that version 1.2 is on the way and promises some great improvements.

Compiling May Not Be for Everybody

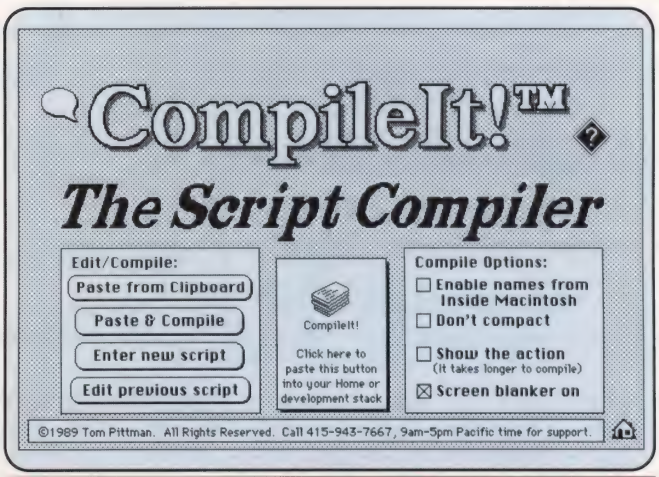
HyperCard was designed to be immediately accessible to even a naive Macintosh user. *CompileIt!* is not. It is, in the words of the manual, "a development tool for experienced HyperTalk scripters." The first thing Heizer's *HyperCard* chief Brian Molyneaux told me as he handed me a copy of *CompileIt!* was "Read the manual." After reading it, which is short but complete, even if it may be little too technical for many *HyperCard* users, I realize that compilers have never been meant "for the rest of us." When something goes wrong with an interpreted program, such as a HyperTalk script, you are generally left staring at a simple error message. You fix it and try again. A compiled program must be designed and written carefully. When something goes wrong with a compiled program written in Pascal or C, you can easily bomb your system and end up looking at a screen full of what looks like refried bits.

To bring compilation to *HyperCard* is not simple. It is to programmer Tom Pittman's credit that he has combined the standard *HyperCard* environment with a time saving compiler written virtually all in HyperTalk.

What *CompileIt!* Is, and What It's Not

CompileIt! is a *HyperCard* tool, like many other *HyperCard* tools. To a sophisticated scripter who wants to selectively speed up scripts that are slow *because they are interpreted*, it can be a powerful addition to a stackware developer's toolkit.

CompileIt! is not the answer, however, to all your HyperTalk speed problems. You don't create instant compiled HyperTalk by opening your latest stack and pushing the magic "Compile" button. In addition, it is not a direct link to SANE (Apple's highly sophisticated Standard Apple Numerics Environment). It only does integer arithmetic—no decimals .There are ways to work around this, but even simple division is a bit complicated.



Product Name: *CompileIt!*
Company Info: Heizer Software
1941 Oak Park Blvd. Suite 40
Pleasant Hill, CA 94523
Price: $99

Version 1.1 Limitations

One limitation is that *CompileIt!* is written in HyperTalk and consequently runs very slowly. Even a trivially short script can take a minute or two to compile. The major limitation of version 1.1 is that only a subset of HyperTalk is supported. Certain very useful commands and keywords are off limits in a *CompileIt!* script. For example, the words `target`, `visible`, `rect`, `result`, `params`, even the word `the` , are simply are not available. Accessing and changing most object properties is also not an option. This means scripts must be tailored to fit—not exactly what the average scripter was dreaming of.

Heizer has good news for us here. Version 1.2—a free upgrade sent automatically to registered users—should be out (you guessed it) by the time you read this. They tell us it will support *all* of HyperTalk's keywords. The only drawback is that this does *not* mean all HyperTalk commands will be sped up. Every time you use a command that requires *HyperCard* itself do something, `go` to a card, `put` or `get` anything, and so on, your compiled script has to ask *HyperCard* just like you do from a regular script. These so called "callbacks" to *HyperCard* take the same amount of time (or longer in some cases) than your regular script. So even removing

Listing 1

A script to test CompileIt! It merges two alphabetized fields that contain many similar entries into a single alphabetized field with no duplications in the final field. Only the function was compiled.

```
on mouseUp
  put field 1 into string1
  put field 2 into string2
  set cursor to watch
  put mergeLists(string1,string2) into mergeString
  put mergeString into field 3
end mouseUp

function mergeLists string1,string2
  put 1 into lineof1
  put 1 into lineof2
  repeat until (line lineof1 of string1) = empty and ¬
    (line lineof2 of string2) = empty
    put line lineof1 of string1 into line1
    put line lineof2 of string2 into line2
    if line1 = line2
    then
      put line1 & return after mergeString
      add 1 to lineof1
      add 1 to lineof2
    else if line1 < line2
    then
      put line1 & return after mergeString
      add 1 to lineof1
    else
      put line2 & return after mergeString
      add 1 to lineof2
    end if
  end repeat
  return mergeString
end mergeLists
```

the limitations to script words doesn't change the fact that *CompileIt!* can only do *some* things faster. The bottom line is that even with the new version of *CompileIt!*, you must tailor scripts to take advantage of what *CompileIt!* does well, and leave out those things where it does not help. This means putting your scripts into different modules and calling the compiled scripts just as you call any XCMD or XFCN.

CompileIt! has always supported direct calls to the Macintosh Toolbox in Read Only Memory (ROM). The advantage of having access to "ROM" calls may be apparent to only a few sophisticated Mac users. The average user might be lost as to how to employ the extremely powerful Macintosh Toolbox. Heizer is planning to release at the same time as Version 1.2 an add-on disk of developer tools. These will help show you how to use these ROM calls to do things not available directly from HyperTalk. One example is a *CompileIt!* script that implements menus. The details of this subject are more suited to our "Xpanding *HyperCard*" column. Suffice it to say, this is a great addition for the sophisticated or adventuresome One must determine for oneself if the effort required to learn about the Toolbox is worth the time involved.

A Little Test

Listing 1 shows a short routine that we wrote and compiled to see what kind of performance improvement *CompileIt!* provides. The task is to merge two fully alphabetized user dictionaries generated from two, different versions of Letraset's page layout software, Ready,Set,Go! These are simple text files easily imported into Fields 1 and 2 of a *HyperCard* stack. The routine simply merges the two into a third alphabetized field, so that there are no duplicate entries in the field. We compiled only the function `mergeLists`. The uncompiled routine takes from 1-1/2 to 4 minutes to operate (depending on the hardware) when merging two lists of 300-400 words each. Compiled, it takes about half that long. We've included it on our *StackSolutions* disk to demonstrate both compiled and uncompiled versions so you can see for yourself the kind of savings that occur in this type of application.

It took about 30 minutes to compile the `mergeLists` handler on a Mac II. We also did it on a 1 megabyte Mac Plus, but after an hour and a half I went to bed. The next morning it was done and worked fine.

In short, *CompileIt!* can save you time if you are willing to spend some time. If you have a handler that takes a long time to operate, you have a choice. You can wait a long time, once, for the handler to compile. Or, you can wait every time you use the handler in your stacks.

—Roger Wood

Guide to *HyperLink Magazine's* Back Issues

Each companion *StackSolutions* disk contains all the software described in the issue and more!

Volume 1, Number 1

April/May 1988

In the Cards
The author of HyperCard's "Help Stack" explores the layers of HyperCard
—Carol Kaebler

HyperCard Tips
HyperLink's idea swap meet
—William K. Balthrop

Short Stacks
A beginner's bonanza—My Diary
—Joan Donaldson

Shafer On Scripting
Featuring a tip on accumulating totals
—Dan Shafer

Xpanding HyperCard
Fool-proof input via the BarButton XCMD
—Lloyd Maxfield

Order Entry DataBase Part One: CustomerStack
A relational database written entirely in HyperCard helps you track your customers
—David G. Brader

Scanning the Arts
Using ThunderScan to input HyperCard Art
—Chris Paapanen

The I Ching
A stack to help guide your future—
Chris King & Roger Wood

Order Entry DataBase Part Two: OrderStack
The conclusion of building a relational database in HyperCard
—David G. Brader

Survey of HyperCard Resources
What is the best source of HyperCard help for you?
—Roger Wood

MacRecorder from Farallon Computing
Record sounds to use directly in your stackware
—Chris King

Volume 1, Number 2

June/July1988

Short Stacks
In praise of HyperCard's easy access to new users
—Joan Donaldson

Shafer On Scripting
Using script abstracting to increase efficiency and handling radio buttons with minimum scripting
—Dan Shafer

Buttons & Bows
A button to help you find the mouse location
—Mark Richardson

In the Cards
The author of HyperCard's "Help Stack" discusses when to put multiple backgrounds into your stacks
—Carol Kaebler

HyperCard Tips
HyperLink's idea swap meet
—William K. Balthrop

Xpanding HyperCard
Building a Sound Library with the ListRes XCMD
—James Paul

Hyper-Neuroanatomy
An educational stack that uses digitized brain sections
—Victor S. Johnston, PhD.

Class Stacks for Educators
A simple to use teacher's aid
—David G. Brader

A Different Approach to HyperTalk
An excerpt from his latest book—Danny Goodman's HyperCard Developer's Guide
—Danny Goodman

Help for HyperCard Printing
Featuring reviews of these printing utilities: Reports from Actwision and HyperCONTROL from Nordic Software
—Roger Wood

Volume 1, Number 3

September/October 1988

Shafer On Scripting
Using script abstracting to increase efficiency and handling radio buttons with minimum scripting
—Dan Shafer

In The Cards
The author of HyperCard's "Help Stack" discusses text manipulation techniques
—Carol Kaebler

HyperCard Tips
A discussion of using "on idle" handlers
—Rbett Savage

Buttons & Bows
A button that proves to be part of a moving experience
—Kenneth S. Hulme

Xpanding HyperCard
Introducing SReplace, a new search and replace XCMD
—James Paul

Short Stacks
An easy to create letter-writing stack featuring push-button fonts
—Joan Donaldson

Budget Print A HyperTalk Utility
There is a tool for doing HyperCard reports that doesn't require extra expense: the HyperTalk scripting language
—Steve Roti

Venturing into Hypertext in The Lands of PC
An overview of Guide, Zoomracks, KnowledgePro, and our own HyperDocuments
—The HyperLink Staff

Adventure in HyperLand
Create your own custom adventure
—William K. Balthrop

HyperExpo and Product News
A report on the first ever HyperExpo and a look at some of the best stackware at the show
—Roger Wood

Volume 1, Number 4

November/December 1988

Short Stacks
Using HyperCard's built-in financial functions to make a Loan Calculator
—William K. Balthrop

Shafer On Scripting
Techniques for doing hypertext with HyperTalk
—Dan Shafer

In The Cards
A stack that helps Apple's office keep track of John Sculley
—Carol Kaebler

HyperCard Tips
Self-reflective HyperTalk or what scripts would M. C. Escher have made?
—Rbett Savage

Xpanding HyperCard
The Pad XFCN in a comparison of HyperTalk and Pascal
—James Paul

What ORACLE Foretells for the Macintosh
Big dividends for developers
—Dan Shafer

InventoryStack
Can your business keep track of its products? Now you can with your own InventoryStack
—William K. Balthrop

HyperCard and the 1-Megabyte Macintosh
Getting the most from a minimum configuration
—John DiBiasi

HyperCard and the Apple Scanner
How to integrate the Scanner with our InventoryStack
—David G. Brader

Reviews of HyperCard Products
A look at 101 Scripts & Buttons, HyperTools, ManorTools, HyperAnimator, and HyperCard Toolkits from APDA
—Roger Wood

Volume 2, Number 1

January/February 1989

Shafer On Scripting
Hypertext in HyperCard revisited; a look at an indexing technique
—Dan Shafer

HyperCard Tips
Your hot tips can earn you money and fame
—Kevin Altis
—James Baggott

Short Stacks
Use this "Personal Financial Statement" stack to calculate your net worth—it may be more than you thought
—William K. Balthrop

Xpanding HyperCard
The HyperCard Toolbox and the glue routines
—James Paul

HyperCard Haiku
The medium is the message
—Rbett Savage

Using ORACLE's HyperSQL
The nuts and bolts of a new Macintosh product
—Dan Shafer

Searching Stacks with Keyword in Context
Expanding HyperCard search capabilities
—Steve Roti

Networking with HyperCard—First Steps
If you think HyperCard and networking are not compatible, read this
—David G. Brader

HyperCard and CD ROM
A look at the present state of this exciting melding of technologies, featuring the BMUG PD-ROM and the Xiphias Time Table of History
—Roger Wood

Reviews of Products
This issue's products include ColorCard, Stack Cleaner, RDAide, and Tax Stacks

Volume 2, Number 2

March/April 1989

Shafer On Scripting
Moving with style among the objects
—Dan Shafer

HyperLink Tips
Updating a "Time" Field
—Rodney Magnuson

Interfacing the Future
The author of 101 Scripts and Buttons for HyperCard *begins a new series addressing the issues of interface design*
—Craig Ragland

HyperCard Haiku
Variable varities—speed and using globals
—Rbett Savage

Short Stacks
Planning your retirement can be as easy as filling in the blanks
—William K. Balthrop

Xpanding HyperCard
An in-depth look at an ICON animating XCMD from the author of Icon Factory
—James Paul

SQL Basics, Part 1
Learning about Structured Query Language (SQL) can help you link to databases outside the Macintosh domain
—Dan Shafer

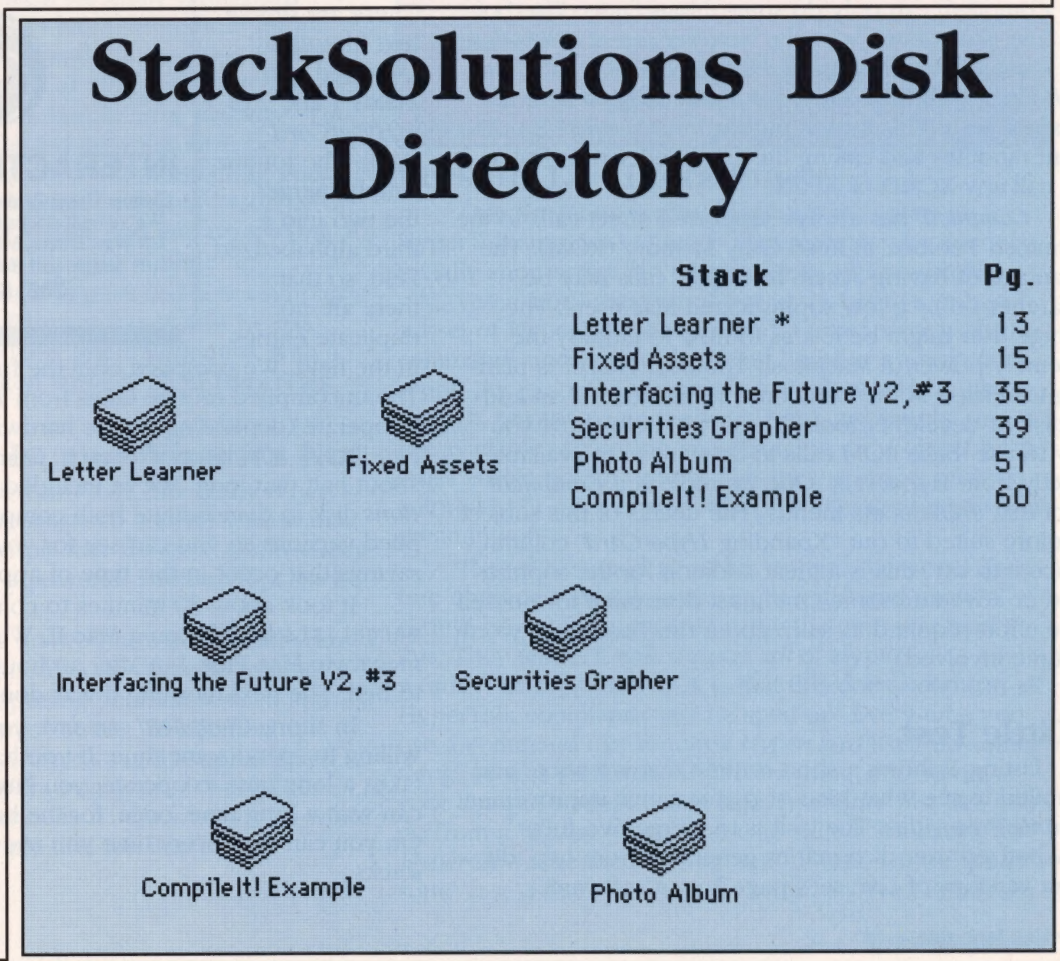
The Big Message Box
Sometimes Hypercard's single-line message box hinders as much as it helps—this is what HyperCard needs
—Kevin Altis

Creating a Selective Mailing List
Here's help from a binary search in Hypertalk
—Larry Walker

SuperCard—A Special Report
A peek inside Silicon Beach's new HyperCard "kissin' cousin"
—Roger Wood

Personal Information Managment Tools
Reviews of Focal Point II, Organizer+, Client, and CONTACTS!

Reviews of Products
Reviews of the Manhole, *and* Hypertext for Beginners



HYPERLINK MAGAZINE

The premier HyperCard information magazine! Subscribe Now!

When you subscribe to **HyperLink Magazine**, take advantage of our special offer to receive the optional companion **StackSolutions** microdisk for your first subscription issue — **FREE!** *

Subscribe TODAY! *Mail this subscription form and payment to:*

HyperLink Magazine, P.O. Box 7723, Eugene, OR 97401.

In a hurry? Place your order by calling our toll-free subscription line at:

1 (800) 544-0339 *Please have your VISA or MasterCard handy.*

SUBSCRIBER NAME _____ PHONE (____) _____

COMPANY _____

ADDRESS _____

CITY _____ STATE _____ COUNTRY_____ ZIP _____

METHOD OF PAYMENT: ☐ VISA ☐ MasterCard ☐ Check (enclosed)

Credit Card Number _____ Expiration Code _____

Signature of Card Holder _____

CHOOSE a HyperLink Magazine (HLM) subscription alone —

☐ One Year $25 ☐ Two Years $45 ☐ Three Years $60 (HyperLink is a bi-monthly publication)

☐ ***Yes**, send the free StackSolutions disk.* Enclosed is the $2.95 Shipping & Handling charge.*

OR choose a combined HLM/StackSolutions Disk subscription —

☐ One Year $67.95 ☐ Two Years $124.95 ☐ Three Years $172.95

(The first StackSolutions disk of the combination subscription is absolutely free!)

ALSO Please send the following back issue products —

HyperLink Magazine back issues:

I enclose $4.95 for each one marked.

☐ Vol.1 #1 ☐ Vol.1 #2 ☐ Vol.1 #3 ☐ Vol.1 #4 ☐ Vol.2 #1

☐ Vol. 2 #2

StackSolutions back issue disks:

I enclose $14.95 (includes S&H) for each one marked.

☐ Vol.1 #1 ☐ Vol.1 #2 ☐ Vol.1 #3 ☐ Vol.1 #4 ☐ Vol.2 #1

☐ Vol. 2 #2 ☐ Vol. 2 #3 (for this issue)

Added Air Shipping Rates

Canada/Mexico, per subscription year

Magazine only	Magazine and Disk
$15	$21

(Back Issues >> HLM + $3.00 and Disk + $3.95)

All other countries, per subscription year

Magazine only	Magazine and Disk
$36	$54

(Back Issues >> HLM + $6.00 and Disk + $3.95)

All orders must be in US funds.

Total amount enclosed for the checked items $ _____

*Please note that the premium (StackSolutions microdisk) has a retail value of $12. If for any reason a refund is to be made on this subscription, the value of the premium and any delivered issues are subtracted from the refund. Also, please note: The $2.95 Shipping & Handling must be paid with your magazine subscription to take advantage of this offer.

Macintosh® In Business

NEW 256-Page Guidebook - Free

The MicroGroup offers a balanced line of high technology and general business products resulting from solving our own productivity and production problems where outside solutions were unavailable. Our all new Spring '89 Guidebook was produced on Macintosh - of course!

Macintosh® In Manufacturing

We believe office and factory automation are mandatory components of customer service. We also recognize the necessity of applying automation as the only solution to our continued growth and profits. The most important lesson we learned is that simplicity and ease-of-use are the only considerations in developing any automation system.

Patented 1stKEY™ Database Software

Regardless of your size, 1stKEY can produce just about anything that you will ever need in a business database system - FAST. 1stKEY can provide MIS the productivity of true relational workload distribution and easy to use reporting across platforms. Now anyone can quickly design systems that can easily interact with other systems - even remote sites. 1stKEY features powerful patented FUZZY Triage FIND™ for performing inexact weighted searches across four fields.

MacBOT™ Robotic Systems

The MacBOT-8 data acquisition and control system was created as a flexible tool to apply the power of Macintosh and HyperCard® to your everyday automation requirements - "Personal Automation." MacBOT robots and robotic components can be applied to push-pull and rotary motions and multiple axis devices that look like their larger counterparts. Complete weighing systems available.

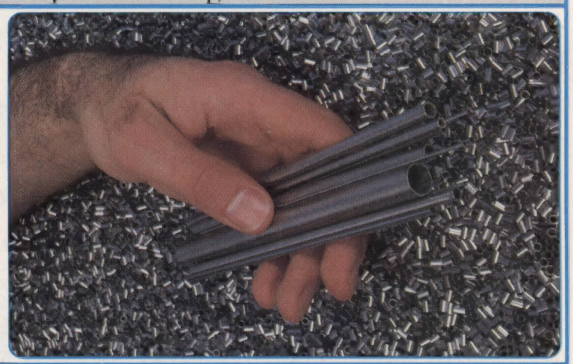
the MicroGroup® Products & Services Specifiers Guidebook

Abridged Pocket Version

Precision Metals

ALL-TUBE® Corporation

Stainless Steel Tubing
Luer and Tube Fittings
Precision Fabrication
Metalworking Tools



Office Automation

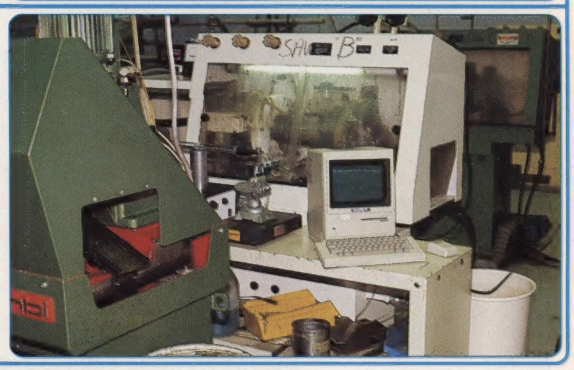
1stDESK® Systems, Inc.

Information Management Systems for the Macintosh®

Factory Automation

MacMOTION, Incorporated

Data Aquisition
Motion Control
Desktop Robots



7 Industrial Park Road Medway, MA 02053 **1-800-255-8823**

Our 256 page Guidebook is also in the >>>THOMCAT<<< Thomas Register Catalog File